2008 - 2009

JAVIER
NÚÑEZ
FERNÁNDEZ

PFC
"GENERADOR DE MELODÍAS BASADO EN REGLAS"

Directores: Tomás de la Rosa Turbides, Sergio Jiménez Celorrio |

TABLA DE CONTENIDOS

1. ÍNDICE DE ÍLUSTRACIONES

1. INTRODUCCIÓN

El presente proyecto, bajo el título *Generador de melodías basado en reglas*, consiste en el desarrollo de un sistema que, a partir de unos parámetros de entrada, genera una melodía musical en formato MIDI.

La aplicación tiene un carácter especial, ya que su principal función es la creación de una composición artística, en concreto una melodía musical, entendiendo por esta una sucesión de notas musicales con un determinado carácter y sentido.

Se aborda este trabajo con la consciencia de la dificultad que entraña la generación de melodías musicales, dada la evidente incapacidad de un ordenador de poder entender o "escuchar" lo que está componiendo. Este factor, aunque intuitivo y simple, se convierte en la principal fuente de dificultad del proyecto, que se ha de subsanar en lo posible mediante la creación de unas reglas estrictas en cuanto a la composición, dejando bastante espacio a la propia aleatoriedad del sistema, para generar melodías claramente diferenciadas entre sí.

Los elementos que contiene (y que se irán explicando) este proyecto son dos principalmente:

- El **sistema experto** sobre el que se definirán las reglas de **generación de las melodías** y la base del conocimiento, que será **CLIPS**.
- El **compilador** escrito en Java (y no perteneciente al desarrollo de este proyecto) que transforma un archivo con la melodía representada en lógica de predicados y lo traduce a formato MIDI.

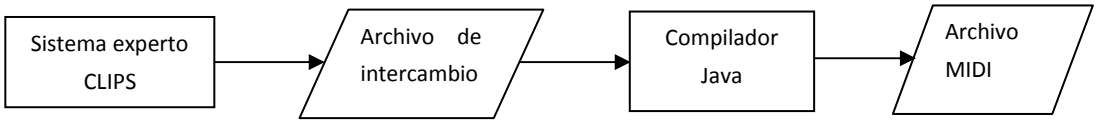


Ilustración 1 – Esbozo de arquitectura del sistema

Basándose en una herramienta de desarrollo de sistemas expertos, como es CLIPS, y en un compilador, éste proyecto realiza las siguientes tareas:

- Definir un lenguaje que permita a un sistema experto la representación de melodías musicales.
- Definir mediante reglas el conocimiento experto que establezca los patrones que normalmente utiliza un músico profesional a la hora de componer una melodía musical (intervalos entre notas, altura de las notas, distancia, cadencias, etc.)
- Una vez creada la melodía, formatearla adecuadamente para ser entrada de un compilador realizado en JAVA que transformará la melodía representada en por el

sistema experto, en una melodía en formato MIDI y, por lo tanto, reproducible en cualquiera de los numerosos reproductores que existen hoy en día.

Supone un reto la creación de un sistema que permita generar una composición artística, dado el carácter de creatividad e imaginación que ello conlleva. Muchos autores desoyen a los teóricos de la música y de la armonía, considerando que la música no es algo que siga unas reglas matemáticas o lógicas, que es algo que nace del propio espíritu del músico y que basarse en unas reglas desvirtuaría la propia esencia de la creación musical.

No estando totalmente en desacuerdo con esta teoría, este trabajo pretende demostrar que, aunque ya se ha hecho en varios programas comerciales conocidos (como es el caso de Band In A Box[1]), es posible, mediante unas reglas de armonía y composición, generar una melodía musical que podría haber sido compuesta por un músico.

En este proyecto, se considera un factor fundamental la aleatoriedad, tanto en la estrategia de búsqueda de soluciones, como en la propia representación. Esa aleatoriedad nos ofrecerá dos vertientes, una positiva y una negativa para el desarrollo del proyecto. La positiva es la generación de melodías claramente diferenciables entre sí; la negativa es decidir qué nota se va a "tocar", teniendo enfrente numerosas posibilidades.

El desarrollo de instrumentos de control de esas posibilidades es sin duda el núcleo del sistema y lo que definirá en última instancia el carácter de la melodía.

[1] Band In A Box es un software para reproducción de MIDI que incluye entre sus funciones algunas como generación automática de melodías o solos de guitarra.
Más información: www.band-in-a-box.com/es/

2. ESTADO DEL ARTE

Este apartado explica la situación actual de las tecnologías y los proyectos similares al presente, tanto en el terreno de la música digital (tecnologías, influencia…), como en el de la inteligencia artificial aplicada a la música digital.

2.1. MÚSICA DIGITAL

La era digital tiene un buen ejemplo de impacto en la música. El 90% de la música que se escucha actualmente es digital, ya sea escuchándola a través de un reproductor portátil, una página web o sintetizada en un concierto en directo.

2.1.1. REPRODUCCIÓN

La reproducción de música hoy en día es principalmente digital. El formato que abrió este camino fue el MP3 (MPEG-1 Audio Layer 3), creado en 1995, y que se convirtió en el formato estándar para *streaming*[2] de audio en internet, ya que conseguía una excelente compresión de la música sin apenas perder calidad.

Otros formatos de reproducción y grabación son:

- **WAV**, un formato de audio digital sin compresión, que mantiene la más alta calidad del sonido, la de un disco compacto.
- MIDI (*Musical Instrument Digital Interface*), del que se hablará de manera extensa en el apartado 4.2. del presente manual.
- RealAudio, que fue un formato que tuvo gran importancia a principios de los 90 por su gran capacidad de compresión, pero al incrementarse el ancho de banda, y dado que el mp3 era un formato libre, reproducible en cualquier reproductor musical, fue perdiendo importancia y presencia.
- AU, que es un formato de audio para UNIX.

En este proyecto nos centramos en el formato MIDI, el cual pertenece a la tercera categoría, que representa la música de forma simbólica y permite, por tanto, la aplicación directa de técnicas de Inteligencia Artificial.

2.1.2. GRABACIÓN

En el mundo de la grabación musical, la era digital ha traído consigo un cambio creciente en este aspecto: donde antes reinaban enormes mesas de grabación analógica, ahora existen ordenadores personales que consiguen realizar el mismo trabajo con mucho menor coste y con una calidad incluso superior.

[2] **Streaming** es un término que se refiere a ver u oír un archivo directamente en una página web sin necesidad de descargarlo antes al ordenador. Se podría describir como *"hacer clic y obtener"*

Existen numerosos programas que permiten la secuenciación y grabación de música, que permiten modificar las pistas añadiéndoles efectos de todo tipo, corrigiendo los ruidos, tratando las ondas musicales, etc.

La música digital ha permitido asimismo que la grabación de una composición musical de calidad, esté al alcance de músicos amateurs, que pueden "colgar" su música en páginas de internet a través del formato MP3. Este formato ha permitido que una canción redujera su tamaño de 40 megas a 4 o 5 sin perder calidad. Esto ha permitido que la difusión de música a través de Internet se vaya incrementando cada vez más.

Estos sistemas de secuenciación y grabación (como por ejemplo Cubase[3] o Protools[4]) también han visto un gran *Dorado* en el MIDI, tanto en los instrumentos como en los secuenciadores, ya que al ser señales digitales, eran perfectamente comprensibles por un ordenador. Los instrumentos MIDI son un gran avance también, ya que con un piano MIDI y un sintetizador, se puede crear, no sólo el sonido de un piano, sino el de un violín, incluso de una batería.

[3] **Cubase** es una serie de aplicaciones informáticas para editar audio digital, MIDI y un secuenciador de música, (comúnmente conocidas como DAW - Digital Audio Workstation), creadas originalmente por la firma alemana Steinberg en 1989. (FUENTE: WIKIPEDIA)

[4] **Pro Tools** es una estación de trabajo de audio digital *(Digital Audio Workstation o DAW, en inglés)*, una multiplataforma de grabación multipista de audio y midi, que integra hardware y software. Actualmente, por sus altas prestaciones, es el estándar de grabación en estudios profesionales, usado mundialmente. (FUENTE: WIKIPEDIA)

2.2. INTELIGENCIA ARTIFICIAL APLICADA A LA COMPOSICION DE MÚSICA

La inteligencia artificial es aplicada a la música en tres vertientes diferentes:

- La composición (1)
- La improvisación (2)
- La interpretación (3)

Ya que tanto la interpretación como la composición musical son adquiridas por el hombre a raíz de la observación, la imitación y el estudio de otras obras musicales, la dificultad principal que se ha de afrontar al aplicar técnicas de inteligencia artificial a la música es la representación del conocimiento en este tema.

2.2.1. SISTEMAS DE COMPOSICIÓN MUSICAL

En cuanto a los **sistemas de composición musical** unos de los pioneros son Hiller y Isaacson (Isaacson, 1993), que, utilizando las cadenas de Markov[5], realizaron la composición de un cuarteto de cuerda, generado a partir de unas reglas heurísticas basadas en la armonía clásica y en *counterpoint,* un término inglés que se refiere a la conjunción de varias melodías, mediantes las cuales se elegía la nota a incluir y en caso de no encontrar ninguna que satisficiera las reglas, empezar un mecanismo de *back-tracking* (o vuelta atrás).

En el 72 Moorer (Moorer, 1993) rompe con la técnica probabilística, aplicando repetición de patrones de notas basadas en progresiones armónicas. Uno de los componentes más importantes de este sistema fue la subdivisión de las reglas de su sistema en **reglas de aplicación** y **reglas de peso**, las primeras indicando qué reglas se deben aplicar y la segunda para medir cuál de ellas es de prioritaria aplicación.

En el 93, Levitt (Levitt, 1993), rehuyendo también de la aplicación de la probabilidad en este campo, aplicó un sistema basado en una descripción de estilos musicales basada en restricciones, éstas últimas denominadas por él mismo como *style templates* o plantillas de estilo.

Otro de los temas de estudio ha sido la armonización de un grave, con el que Ebcioglu (Ebcioglu, 1993) en 1993 hizo una gran aportación con su programa CHORAL, que armonizaba una melodía con el estilo de Johann Sebastian Bach, usando reglas heurísticas, cuya principal cualidad fue la de contemplar un conjunto de primitivas lógicas como la vista de los acordes, la

[5] Cadena de eventos, en los cuales la probabilidad de que ocurra un evento depende directamente del evento que se haya producido anteriormente.

vista de la melodía, etc. Otros estudios, como los de Bharucha (Barucha, 1993) o Feulner (Feulner, 1993), se basaron en la utilización de redes neuronales para aproximarse a la resolución de la armonización.

En 1998, Sabater y Arcos (Sabater & Arcos, 1998), elaboran un sistema basado en reglas y en el CBR (*Case-Based Reasoning* o razonamiento basado en casos). Un sistema basado únicamente en reglas cojea por el hecho de que las reglas no deben generar la música sino que ha de ser la música la que genera las reglas. De ahí la apuesta de los autores por extraer la información que se perdería a partir de armonizaciones realizadas de manera real para el aprendizaje del sistema.

Otros proyectos a destacar en este terreno son la elaboración de música a partir de sensores en un bailarín, de Morales y Manzanares (Morales - Manzanares, 2001). También el proyecto EMI de David Cope (Cope, 1987), que trataba de imitar los métodos de composición de los clásicos.

2.2.2. SISTEMAS DE IMPROVISACIÓN MUSICAL

Este tipo de sistemas nacieron con posterioridad a los de composición, dada la complejidad mayor de los métodos heurísticos.

Uno de los primeros en implementarlo fue Fry (Fry, 1993), con su sistema *Flavors Band* construido en un lenguaje embebido en LISP. A partir de una entrada y mediante unas funciones aleatorias y unas restricciones musicales, el sistema de Fry conseguía, a partir de la definición de un estilo, construir una improvisación.

Papadopoulus y Wigins (Papadopoulos, 1998), así como Biles (Biles, 1994) hicieron uso de algoritmos genéticos para construir sus sistemas de improvisación. El del segundo utilizaba a un humano para calificar las improvisaciones, mientras que el del primero tenía su propio sistema de calificación.

2.2.3. SISTEMAS DE INTERPRETACIÓN MUSICAL

Los dos sistemas vistos con anterioridad no tenían como centro la expresividad, cualidad ésta necesaria en la elaboración de un sistema de interpretación musical, dado el interés principal del cerebro por el cambio. Este hecho ha sido estudiado y se ha demostrado que ante sonidos similares o repetición de los mismos, las neuronas ven reducidas su actividad, mientras que la aumentan ante alteraciones en el ritmo, el tempo o la secuencia de las notas. Estas alteraciones pueden explicar porqué la música realizada por un humano a través de un instrumento, es más valorada que la música creada por un sintetizador; un instrumento real envía a la corteza cerebral de la audición más estímulos que una música sintetizada.

Uno de los primeros intentos de encontrar esta expresividad a través de un sistema basado en reglas fue Johnson (Johnson, 1992), que desarrolló un sistema experto que era capaz de determinar el tempo y la articulación de una pieza de Bach. Las reglas fueron elaboradas por dos humanos expertos. Los resultados fueron buenos, su principal limitación era que solo funcionaba para fugas escritas en compás de tipo 4x4 de Johann Sebastian Bach, pero también sentaba las bases de una posible generalización de su sistema.

El trabajo del grupo KTH de Estocolmo es uno de los sistemas de interpretación más conocidos. Su sistema incorporaba reglas para tiempo, dinamismo y articulación con MIDI. Las reglas fueron inferidas de la teoría musical sobre todo usando el método de aproximación análisis por síntesis (analysis-by-syntesis). Estas reglas eran de tres clases, las **reglas de diferenciación**, que implementaban las diferencias entre tonos de escalas, las **reglas de agrupación**, que establecían que tonos debían ir juntos y las **reglas de ensamblado**, que sincronizaban las diferentes voces.

Canazza (Canazza, 1997) hizo una aportación novedosa desarrollando un sistema que analizaba cómo un músico expresaba determinadas intenciones reflejándolas en una interpretación. El estudio arrojó dos dimensiones diferentes de expresión: una relativa a la energía y otra a la cinética.

Otra aproximación fue el uso de las técnicas de redes neuronales de Bresin (Bresin, 1998) con un sistema que combinaba reglas de decisión con las redes neuronales. La salida de las redes neuronales expresaban las desviaciones de tiempo e intensidad de sonido.

La principal limitación de los sistemas basados en reglas, es la dificultad para encontrar reglas generales, suficientes para capturar la variedad presente en las diferentes interpretaciones de una misma pieza, interpretada por el mismo músico. Además, cuantos más recursos expresivos se pretenden incluir en un modelo, mayor es la dificultad de creación de las reglas apropiadas para ellos.

3. OBJETIVOS DEL PROYECTO

En este punto se abordan las ideas generales sobre el alcance del proyecto y lo que se pretende investigar.

El proyecto tiene como objetivo principal conseguir generar una melodía musical a partir de unas reglas que se establecen en un sistema de producción, en el caso del presente trabajo, la herramienta CLIPS.

Desgranando este objetivo, se pueden ver los siguientes objetivos derivados que se pretenden alcanzar:

- Representación, de la manera más cercana, de una melodía musical en un sistema de producción.
- Definición de unas reglas que permitan al sistema ir componiendo una melodía, teniendo en cuenta las restricciones derivadas del estudio musical.
- Traducción de una melodía representada como hechos CLIPS, generada a través de un compilador JAVA a formato MIDI.

4. DESARROLLO

Este capítulo incluye todos los aspectos técnicos y de desarrollo del proyecto. Dentro del mismo se encuentra la evolución de la ejecución del proyecto así como aspectos de arquitectura, desarrollo y elementos que se han utilizado para conseguir llevarlo a cabo.

4.1. TEORÍA BÁSICA

En esta sección se presentan definiciones teóricas, que sirven de base para la comprensión del resto de secciones de este punto.

4.1.1. LA MÚSICA MIDI

Las siglas MIDI vienen de ***Musical Instrument Digital Interface***. El desarrollo de los sistemas MIDI ha sido el mayor catalizador de la reciente explosión en la tecnología musical. El MIDI ha permitido que los músicos pudieran hacer uso de sonidos para sus grabaciones, que quizá no estaban a su alcance por la falta de conocimiento de los instrumentos.

Existen tres grandes grupos de dispositivos que se utilizan para manejar la música MIDI:

- **Controladores (Drivers)**: Transforman la interpretación musical en mensajes MIDI para ser transmitidos. Dentro de este grupo estarían, entre otros, los instrumentos MIDI, que no son más que instrumentos cuya salida no es música, sino un grupo de bits, que interpretados en un sintetizador se convertirán en música.

- **Secuenciadores**: Graban, editan y reinterpretan la música almacenada. Los secuenciadores normalmente permiten tanto el procesamiento de audio como de MIDI. Los secuenciadores pueden ser analógicos o digitales, aunque normalmente se utilizan los digitales. Un ejemplo de secuenciador podría ser, por ejemplo, el conocido Steinberg Cubase, que permite el uso, tanto de pistas de MIDI, como de audio.

- **Sintetizadores:** Reciben la información MIDI y la traducen al sonido final de los instrumentos. Son reproductores que interpretan los paquetes de bits que le llegan y los reproducen en forma de música.

Una comunicación MIDI consiste en paquetes de varios bytes, que empiezan por un *byte de estado,* seguido por uno o dos bytes de datos.

Los bytes de estado comienzan con un 1 (1xxx xxxx) mientras que los bytes de datos empiezan con un 0 (0xxx xxxx). Cada byte está rodeado de un bit de comienzo y un bit de parada, haciendo que cada paquete ocupe 10 bits. Los mensajes tienen el siguiente formato:

- **Mensajes de canal**

- **Canal de voz**: Control las 16 voces de los instrumentos (timbres, patches), reproduce notas, envía datos de control, etc.
- **Canal de modo**: Define las respuestas del instrumento ante mensajes de voz, envío sobre el canal básico entre instrumentos.

- **Mensajes de sistema**
  - **Comunes de sistema**: Mensajes orientados a todas las redes de instrumentos y dispositivos.
  - **De sistema en tiempo real**: Orientados para sincronizar todos los instrumentos y dispositivos contenidos en una red. Contiene solo bytes de estado y se usa para la sincronización entre dispositivos (donde se hace esencial un reloj de tiempo)
  - **Exclusivos del sistema**: Orientados originalmente a los fabricantes, se han expandido para incluir codificación de tiempo MIDI, etc.

CANAL	**VOZ**	Nota activada Nota desactivada Postpulsación monofónica Postpulsación polifónica Controles Cambio de programa Variación de tono
SISTEMA	**MODO**	Omni ON / OFF Poly ON / OFF Local ON / OFF All Notes OFF
	COMÚN	Posición en la canción Selección de canción Afinación
	TIEMPO REAL	Pulso de reloj Inicio Pausa Final Active Sensing System Reset
	EXCLUSIVO	Sistema exclusivo

Ilustración 2 - Formato de mensajes de MIDI

MENSAJES DEL CANAL DE VOZ

La mayor parte de los dispositivos MIDI están equipados para recibir mensajes MIDI en uno o más de sus 16 números de canal. Una voz de un dispositivo (o parche, programa, timbre) responderá a mensajes enviados por el canal en el que está conectado e ignorará todos los otros mensajes de canal, análogamente a una televisión que solo recibe la señal de la cadena que se quiere ver y se ignoran las otras.

La excepción a esto es el modo OMNI. Un instrumento establecido en el modo OMNI de recepción, aceptará y responderá a todos los mensajes de canal, con independencia del número de canal.

Los mensajes MIDI más comunes son los **mensajes de canal de voz** listados en la tabla inferior. Transmiten información de cuándo cambiar una nota a encendida o apagada, a qué parche cambiar, etc.

CÓDIGOS DE MENSAJES DE VOZ					
Byte de estado	**Byte de datos 1**	**Byte de datos 2**	**Mensaje**	**Leyenda**	**Descripción**
1000nnnn	0kkkkkkk	0vvvvvvv	**Nota OFF**	**n** = canal **k** = nota # 0-127 **v** = velocity (0-127)	Se utiliza para **detener una nota**. El byte de estado indica el canal, el primer byte de datos la nota que se va a apagar y el segundo la velocidad, que corresponde a la velocidad con que se suelta la tecla
1001nnnn	0kkkkkkk	0vvvvvvv	**Nota ON**	**n** = canal **k** = nota # 0-127 **v** = velocity (0-127)	Se utiliza para **activar una nota**. El byte de estado indica el canal, el primer byte de datos la nota que se va a encender (el código 60 es el Do central) y el segundo la velocidad, que corresponde con la velocidad con que se pulse la tecla y que, según el instrumento con que se trabaje, puede suponer un incremento de la intensidad o no.
1010nnnn	0kkkkkkk	0ppppppp	**After Touch (Presión de las teclas)**	**n** = canal **k** = nota # 0-127 **p** = pressure (0-127)	Se tiene en cuestión la presión de la tecla a la hora de reproducir el sonido.
1011nnnn	0ccccccc	0vvvvvvv	**Control de cambio**	**n** = channel **c** = controller **v** = controller value(0-127)	Los veremos posteriormente, son los controladores.
1100nnnn	0ppppppp	[none]	**Cambio de programa**	**n** = channel **p** = preset number (0-127)	Cambia el programa del instrumento, es decir, cambia el sonido (guitarra, piano, viola, etc.)
1101nnnn	0ppppppp	[none]	**Canal de presión**	**n** = channel **p** = pressure (0-127)	
1110nnnn	0ccccccc	0fffffff	**Pitch Bend (Bending)**	**n** = channel **c** = coarse **f** = fine (c+f = 14-bit resolution)	Desplazan la altura tonal de la nota hacia arriba o hacia abajo. Sirve para desafinar el sonido, simulando así el estiramiento de las cuerdas de una guitarra, o similar

Ilustración 3 - Códigos de mensajes de voz

Por ejemplo, **si tocamos la tecla do** de un piano se mandaría la siguiente información:

1001xxxx (note on)

00111100 (valor 64, que corresponde a la nota do)

0xxxxxxx (la velocidad con la que haya sido apretada la tecla)

Pero **al soltarla**, se puede omitir el byte status y apagarla por volumen (otra posibilidad es que usase el 1000xxxx - note off - para apagarla), es decir, transmitiría sólo los dos siguientes bytes:

00111100 (valor 64 que corresponde a la nota do)

00000000 (la velocidad cero, que indica que tiene que dejar de sonar esa nota)

Eventos simultáneos en MIDI deben ser enviados como una cadena de comandos de manera serial. Un acorde de tres notas, por ejemplo, será transmitido como 3 pares de notas sincronizadas. Debido a la velocidad de transmisión de 31.25 Kilo baudios, es percibido como simultaneo, aunque en realidad estén iniciándose de manera serial. Sin embargo, dado que los instrumentos polifónicos han incrementado el número de voces, en ocasiones se pueden producir indeseables arpegios o errores de datos.

MENSAJES DEL CANAL DE MODO

Los mensajes de canal de modo están dedicados a controlar todas las funciones de **todos** los canales de voz de un instrumento. La especificación MIDI 1.0 supone que los instrumentos son diseñados para operar solo en un modo en cada tiempo. Los mensajes de canal de modo usan los números de controlador 122-127 y son enviados a un **canal básico** dedicado.

Existen tres **tipos de modos de canal**:

- **OMNI** → Si está encendido, establece todas las voces del instrumento para responder a todos los mensajes de canal de modo recibidos, ***sin importar el canal a que se envía.***

- **MONO/POLY** → Los modos MONO/POLY modifican las capacidades polifónicas de una voz singular o de múltiples voces. **Un nuevo mensaje NOTE ON en modo MONO finaliza la nota anterior y comienza la nueva nota**. Se podría desear establecer la sintonización a MONO si quieres usar un parche con **portamento** (un glissando[6] entre notas). Un mensaje de canal de modo se asemeja a la siguiente estructura:
  1011nnnn 0ccccccc 0vvvvvvv ***donde***
  n = canal básico
  c = números de controlador 122-127

[6] En música, un **glissando** (remedo del italiano que proviene del francés *glisser*, 'resbalar', 'deslizar') es un efecto sonoro consistente en pasar rápidamente de un sonido a otro haciendo oír todos los sonidos intermedios posibles (no sólo los tonos y semitonos), según la característica del instrumento.

v = acción para ese modo (on/off).

Status Byte	Data Byte 1	Data Byte 2 (& 3)	Description
1011nnnn	01111010 (122)	00000000 (0) = off 01111111 (127) = on	CONTROL LOCAL
1011nnnn	01111011 (123)	00000000 (0)	Todas las notas **OFF**
1011nnnn	01111100 (124)	00000000 (0)	Omni Mode OFF (all notes off as well)
1011nnnn	01111101 (125)	00000000 (0)	Omni Mode ON (all notes off as well)
1011nnnn	01111110 (126)	0mmmmmmm (m=nº de canales)	Mono Mode ON (poly mode off, all notes off as well)
1011nnnn	01111111 (127)	00000000 (0)	Poly Mode OFF (Mono Mode OFF, all notes off as well)

Ilustración 4 - Bytes de estado

Existen 4 posibles combinaciones de modo OMNI-MONO/POLY:

MODE	OMNI	POLY/MONO	RESULT
1	ON	POLY	Los mensajes se reciben por todos los canales de todas las voces
2	ON	MONO	Los mensajes se reciben por todos los cantales, pero el control es monofónico.
3	OFF	POLY	Los mensajes se reciben en un único canal específico, pero permite sonido polifónico.
4	OFF	MONO	Los mensajes se reciben en un canal específico y de manera monofónica

Ilustración 5- modos del MIDI

- **ALL NOTES OFF** → Es útil para programas de secuenciación, donde los mensajes perdidos deberían dejar una nota "colgando" o sonando indefinidamente porque se pierde el comando NOTE OFF.

- **LOCAL CONTROL** → (on/off) Está diseñado para separar la función de teclado de un sampler o sintonizador, de su propia capacidad de producir sonido. Es útil cuando quieres solo notas devueltas por el ordenador para causar el sonido del instrumento, evitando el problema de notas duplicadas. Las notas y otros datos son todavía enviados al puerto MIDI OUT. Normalmente, esto está mejor establecido en el propio instrumento que a través de MIDI.

MENSAJES COMUNES DE SISTEMA Y DE TIEMPO REAL

Los **mensajes de sistema** no incluyen números de canal y están dedicados a direccionar (o mapear) todos los dispositivos del sistema.

Solo existen unos pocos **mensajes comunes de sistema** que se trata principalmente con los secuenciadores de instrumentos de mesa o máquinas de percusión, las cuales deberían haber pregrabado secuencias MIDI. Los mensajes comunes de sistema tienen un byte de estado que empieza con 11110001 (240) y uno o más bytes de datos.

Un **puntero de posición de canción** es un registro que cuenta el número de tiempos desde el inicio de la canción (secuencia). Otros mensajes comunes de sistema, seleccionan que canciones reproducir así como una "petición de tono" para sintetizadores analógicos para afinar sus oscilaciones. Dado el enorme número de tiempos, el puntero de posición de canción usa dos bytes de datos para un mensaje de 14 bits.

Los mensajes de sistema de tiempo real incluyen un reloj de tiempos, que envía 24 'tiempos de reloj' cada cuarto de nota (dependiendo del tempo establecido, por lo que es un tiempo relativo). Los tiempos de reloj del secuenciador interno de un instrumento pueden asimismo controlar las tasas de LFO´s y otros parámetros (por ejemplo, en el Kurzweil K2xxx series, si la fuente de reloj se deja en el interno, los parámetros A Clock y B Clock usarán el secuenciador tempo).

MENSAJES EXCLUSIVOS DE SISTEMA

Los **mensajes exclusivos de sistema** (o SysEx) expanden la funcionalidad del MIDI en varios caminos. Primeramente, esta clase de código fue usada principalmente por funciones de **edición / librería**. Los grandes bancos de patch podrían ser almacenados en una computadora a través de códigos SysEx y devueltos al instrumento cuando sea necesario.

Un código exclusivo de sistema comienza con 11110000 (240 en decimal, F0 en hexadecimal), seguido del identificador del fabricante, después de un número no especificado de bytes de datos de cualquier rango entre 0 y 127 y termina con 11110111 (247 en decimal o F7 en hexadecimal), lo que significa el mensaje de fin de Mensaje Exclusivo de Sistema.

Algunas de las **extensiones** de los mensajes exclusivos de sistema son:

- **MIDI TIme Code**

Como se mencionó anteriormente, el reloj MIDI, es un dispositivo de tiempo relativo, ya que dependen del tiempo musical o **tempo**. Una necesidad real desarrollada para sincronizar MIDI con Video, Dispositivos de Audio, dispositivos de video.

La clase de código utilizada para Código de Tiempo MIDI es denominada Tiempo Real Universal Exclusivo de Sistema. Un mensaje simple se encierra entre el paquete mencionado más arriba y es similar a esto (listado en hexadecimal)

F0 7F 7F 01 00 hr mm sc fr F7

hr = hora y tipo (0 yy zzzzz)
yy = frames por segundo (00=24 fps, 01=25 fps, 10=30 fps, 11=30 fps-drop)
zzzzz = horas 00-17

mn = minutos
sc = second (0-59)
fr = frames (0-29)

La tasa de "frames" es una cuestión delicada. Es importante para cualquiera que quiera sincronizar con una cinta de video que se sepa la tasa de "frames" a la que fue codificada la cinta para establecer (de manera que coincidan) la tasa de "frames" del dispositivo MIDI. Afortunadamente la capacidad de almacenamiento de las computadoras actuales permite a los compositores importar los archivos de video directamente a sus secuenciadores MIDI, eludiendo la necesidad de esta función.

4.1.2. LOS SISTEMAS EXPERTOS BASADOS EN REGLAS

Un sistema experto basado en reglas es una herramienta de desarrollo enmarcada en las herramientas de inteligencia artificial, que se basa en una ejecución no lineal construida sobre:

- Una **base de hechos**, que contiene un conjunto de literales (llamados **hechos**), que definen la situación particular en que se encuentra la ejecución.
- Una **base de reglas**, que contiene un conjunto de reglas de tipo IF-THEN con un antecedente y un consecuente y que *se pueden disparar* cuando el antecedente coincide (es equiparable) con algún "elemento" de la base de hechos.
- Una **agenda**, en la que se almacenan todas las reglas equiparables.

El orden en el que se ejecutan las reglas equiparables que se encuentran en la agenda, depende de la estrategia establecida en el ***motor de inferencia***, que es la parte de un sistema experto que se encarga de resolver el ***conjunto conflicto***, que es el conjunto de reglas equiparables de las que ya se ha hablado con anterioridad, dependiendo de la estrategia establecida. Existen numerosas estrategias de resolución del conjunto conflicto entre las que podemos nombrar:

- *Primera regla*. Esta estrategia consiste en ejecutar antes las reglas que se han generado antes
- *Más conocimiento*. Consiste en ejecutar antes las reglas que equiparan mayor número de variables en el antecedente.
- *Más prioridad.* Es aquella estrategia que resuelve antes las reglas de mayor prioridad, que se establece en cada regla de manera individual.
- *Más específica*. Ejecuta antes las reglas más específicas, es decir, aquellas que contienen mayor número de elementos en su antecedente.
- *Más general*. Ejecuta antes las reglas más generales, es decir, aquellas que contienen menor número de elementos en su antecedente.
- *Elemento más nuevo*. Ejecuta antes las reglas creadas más recientemente.
- *Aleatoriamente*. Ejecuta las reglas de manera aleatoria, es decir, escogiendo una cualquiera al azar.
- *Meta reglas*. Reglas de diferentes tipos con un propósito específico.
- *Mezcla de estrategias*. Esta estrategia hace uso combinado de varias de las reglas disponibles.

"El sistema CLIPS implementa las siguiente estrategia de resolución de conflictos:

1. Las reglas más recientemente activadas se colocan encima de las reglas con menor prioridad, y debajo de las de mayor prioridad
2. Entre reglas de la misma prioridad, se emplea la estrategia configurada de resolución de conflictos
3. Si varias reglas son activadas por la aserción de los mismos hechos, y no se puede determinar su orden en la agenda según los criterios anteriores, se insertan de forma arbitraria (no aleatoria)

4.1.3. ARMONÍA MUSICAL

En este apartado se explican algunas nociones básicas de música, que son fundamentales para entender el resto de la memoria de este proyecto. Para ello, explicaremos la idea general de la armonía musical moderna y la música, para luego explicar cada uno de los conceptos que irán apareciendo en esta memoria.

La música se construye a partir de sonidos musicales. En la música occidental moderna, estos sonidos se han catalogado en 7 notas fundamentales, que son las que forman la escala diatónica:

Do Re Mi Fa Sol La Si

A estas notas se añaden las notas cromáticas, que son las notas que se encuentran entre algunas de las notas fundamentales o diatónicas:

Do Do# Re Re# Mi Fa Fa# Sol Sol# La La# Si Do

Cada nota musical se identifica por tres características fundamentales:

- La **duración** (que depende de la figura con que se represente dicha nota), que define el tiempo que una nota está sonando.
- La **altura** (que depende de la posición vertical en el pentagrama), que define la altura del sonido de una nota (C, D, F#, etc.), es decir, si la nota es más grave o más aguda.
- La **octava,** que en realidad es también la altura de la nota, pero que es necesaria representar aparte, por el hecho de determinar a qué altura se encuentran las notas de diferentes octavas.

En este proyecto se ha utilizado la notación musical norteamericana para las notas musicales, de tal manera que Do se representa con una C, Re con una D… y Si con una B.

A continuación vamos a ver algunos de los elementos fundamentales de la armonía, que serán de ayuda para entender la memoria de este proyecto:

INTERVALOS

Un intervalo caracteriza la distancia en altura entre dos notas dentro de una escala, de tal manera que, si por ejemplo, tenemos la escala mayor diatónica,

Do Re Mi Fa Sol La Si

Entre Do y Re tendríamos un intervalo *de segunda* (distancia de un tono)
Entre Do y Si tendríamos un intervalo *de séptima* (distancia de un cinco tonos y medio)
Entre Mi y Sol tendríamos un intervalo *de tercera menor* (distancia de un tono y medio)

Los intervalos caracterizan tanto los acordes como las melodías y son la base de la armonía musical.

ESCALAS

Una escala es un conjunto de notas que se caracteriza por los intervalos que contiene. El ejemplo más común es el de la escala mayor, que se caracteriza porque existen unos intervalos predefinidos y una distancia entre notas, de tal manera que existe una distancia de un tono entre todas las notas excepto entre 3ª y 4ª y 7ª y 8ª (1ª).

Las escalas son la base para aprender a armonizar, componer, e improvisar en cualquier instrumento.

Una escala, por ejemplo, es la base para la formación de los acordes para una armonización, que se construyen a partir de intervalos sobre las notas de la propia escala.

Existen más de 500 escalas diferentes (de Jazz, Blues, Pop, Country, etc), caracterizadas cada una de ellas por una distancia interválica diferentes entre sus notas y por su sonido.

NOMBRES DE LOS GRADOS DE LA ESCALA

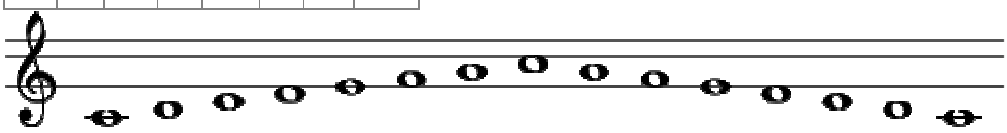
I: tónica
II: supertónica
III: mediante
IV: subdominante
V: dominante
VI: submediante o superdominante
VII: sensible si está a medio tono (semitono) de la tónica (I grado) o subtónica si está a un tono.

ESCALA DIATÓNICA

Es la formación de una escala a partir de las distancias de tono y semitono. La mayoría de ellas están formadas por siete notas, pero las hay también de seis u ocho.

Ordenadas las notas así: *do, re, mi, fa, sol, la, si,* y al añadirle un octavo sonido, de nuevo *do*, hemos formado una escala diatónica.

I	II	III	IV	V	VI	VII	VIII
do	re	mi	fa	sol	la	si	do



ESCALA CROMÁTICA

Es la sucesión de los doce semitonos contenidos en una octava, de los cuales siete son naturales y cinco alterados, lo que hace necesario el uso de la enarmonía.

La enarmonía es el uso de distintos nombres para nombrar las mismas notas, según sean consideradas desde una escala u otra. Por ejemplo: en la escala de *fa#* mayor, la nota que se escribe como *la#* genera el mismo sonido que el *sib* de la escala *fa* mayor. En la música dodecafónica, la escala cromática se utiliza de manera que sea imposible reconocer una tónica

ESCALAS EN LOS MODOS MAYOR Y MENOR

Las escalas más comunes en Occidente suelen ser dos modos: el modo mayor y el modo menor.

ESCALA EN MODO MAYOR

En la escala en modo mayor, los tonos están entre los grados

- I y II
- II y III
- IV y V
- V y VI
- VI y VII

Los semitonos, en cambio, separan a los grados

- III y IV
- VII y VIIII

La escala diatónica es una escala en modo mayor. Cualquier escala mayor que utilice una nota diferente de *do* como tónica debe construirse con este patrón de tonos y semitonos entre sus grados.

ESCALA EN MODO MENOR

En la escala en modo menor, los tonos están entre los grados

- I y II
- III y IV
- IV y V
- VI y VII
- VII y VIII

Los semitonos, en cambio, separan a los grados

- II y III
- V y VI

Esta escala está basada en el modo menor natural, ya que cuando una obra musical está escrita en modo menor (clásica o no) se suelen utilizar simultáneamente varios modos menores: menor natural, menor armónica, menor melódica y menor dórica.

La escala menor armónica es igual a la escala menor natural salvo que debemos elevar un semitono el VII grado. La escala menor melódica es igual a la escala menor natural salvo que debemos elevar un semitono VI y VII grados. La escala menor dórica es igual a la escala menor natural salvo que debemos elevar un semitono el VI grado.

MELODIA

Se llama melodía musical a una secuencia de notas que dotan de un carácter especial a una composición y que le dan su carácter.

La melodía parte de una base conceptualmente horizontal, con eventos sucesivos en el tiempo y no vertical, como sería en un acorde donde los sonidos son simultáneos. Dicha sucesión horizontal puede contener cierto tipo de cambios y aún ser percibida como una sola entidad. Concretamente, incluye cambios de alturas y duraciones, y en general incluye patrones de cambio".

4.1.4. API JAVA-MIDI

Java dispone de un API para el tratamiento digital del sonido que permite desde la reproducción, la grabación o el muestreo hasta la creación y alteración de música MIDI. Este API se compone de cuatro paquetes, de los que nos interesan los dos segundos, que son los dedicados al MIDI:

- javax.sound.sampled
- javax.sound.sampled.spi
- **javax.sound.midi**
- **javax.sound.midi.spi**

Estos paquetes están formados por clases que permiten la manipulación de los sonidos; siendo los dos primeros orientados a la reproducción, mezcla y manipulación de audio y los dos siguientes a la manipulación y creación de archivos MIDI.

Los dos últimos paquetes aportan clases que implementan las siguientes funciones:

- Creación de una secuencia de audio (con la clase **sequence**), que permite entre otras cosas, definir la frecuencia de frames por segundo, el tempo de la canción.
- Creación de una pista de audio (***sequence.createTrack()***)
- Definir un mensaje MIDI, con la clase ***ShortMessage***, que define un mensaje de al menos 2 bytes de longitud.
- Permite añadir al ***track*** un ***ShortMessage*** *(añadir a la pista de MIDI un evento de nota)*
- Etc.

En definitiva, javax.sound es un conjunto de paquetes que permiten abstraer los elementos más técnicos de la creación de un archivo MIDI, y que permite centrarse únicamente en la forma de elaborar las canciones y no en cómo se elaboran.

4.2. AQUITECTURA

La arquitectura del sistema será la siguiente:

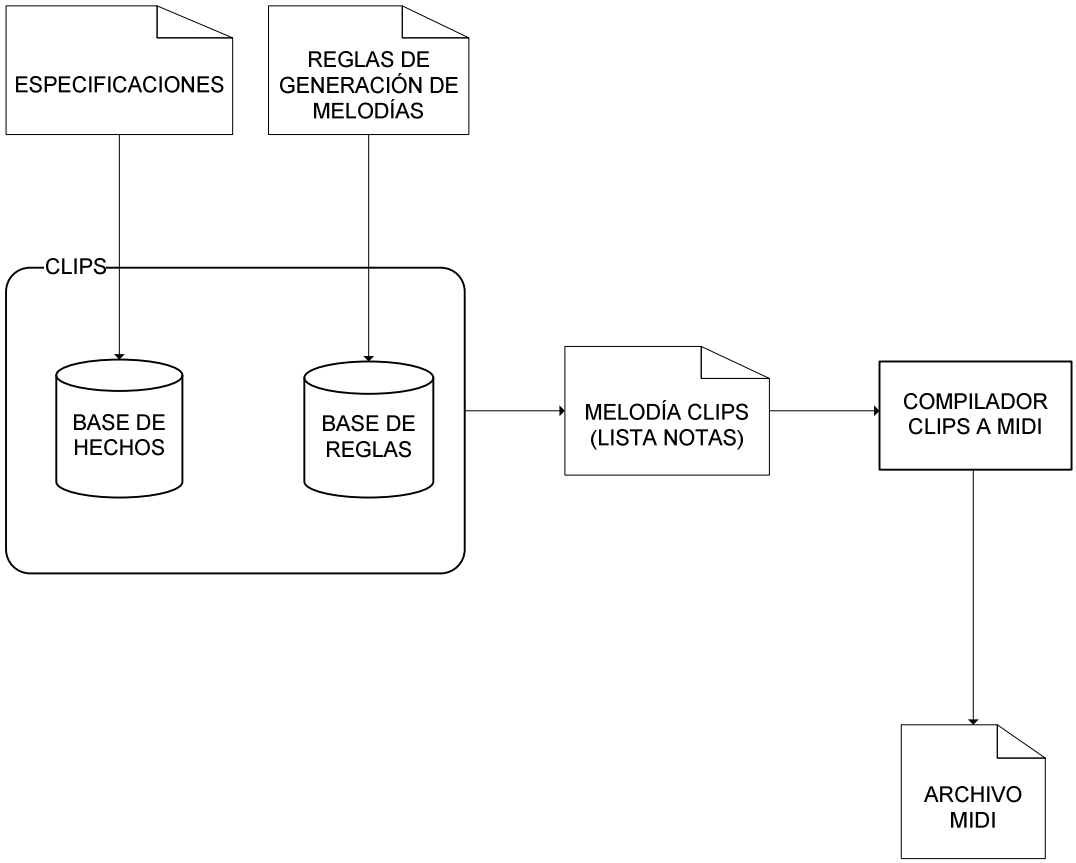


***Figura 6**: Arquitectura del sistema.*

Vamos a ver una breve descripción de los elementos que componen el sistema:

- **ENTRADAS**
  - **ESPECIFICACIONES**: Será un archivo de extensión .clp de entrada al sistema CLIPS que tendrá las especificaciones iniciales (hechos) necesarios para proporcionar al sistema los elementos necesarios para la generación de la melodía.

  - **REGLAS DE GENERACIÓN DE MELODÍAS**: Serán las reglas que permitirán ir generando la melodía, teniendo en cuenta las restricciones a la hora de transitar entre notas.
    Se tratará de que sean lo más generales posibles.

  - **BASE DE HECHOS**: Contendrá tanto los hechos de iniciación, como los que se irán generando para la melodía final; así como hechos de control que nos servirán para conocer el estado en que se encuentra la melodía.

- **BASE DE REGLAS**: Contendrá las reglas que se dispararán para la generación de los hechos de nota.

- **MELODÍA CLIPS**: Será el resultado de la ejecución del programa CLIPS a partir de las especificaciones, que arrojará un fichero con la secuencia de notas que componen la melodía y que pasará al compilador.

- **COMPILADOR CLIPS A MIDI**: Recuperará un archivo con una lista de notas, con sus características particulares, y lo traducirá al formato MIDI, de manera que finalmente pueda ser reproducido en un reproductor estándar.

• SALIDA

- **ARCHIVO MIDI**: Será un archivo con formato MIDI estándar que podrá ser reproducido en cualquier reproductor que soporte este tipo de formato.

4.3. MODELO DE CONOCIMIENTO

PARTITURA MUSICAL

Una parte del modelo de conocimiento se ha extraído directamente de la estructura de una **partitura musical**, que contiene toda la información relativa a una melodía o a una composición musical, por lo tanto se han representado los siguientes elementos:

- **Armadura**: Es la representación de la armadura de un pentagrama musical. De la armadura se ha representado el tiempo de la composición, no se ha tenido en cuenta ni la clave (porque en MIDI no hay claves) ni el número de sostenidos y bemoles, ya que eso será controlado por la escala que se incluya en las especificaciones.

```
(deftemplate armadura
      (slot duracion
            (type INTEGER)
            (allowed-values 2 3 4)
      )
      (slot numero
            (type INTEGER)
            (allowed-values 2 3 4)
      )
)
```

  En este caso, la duración se corresponde con el tipo de tiempo del compás (negra, blanca, corchea, etc.) y el número se corresponde con el número de tiempos por compás.

- **Nota**: La unidad fundamental de la música, con su altura, octava y duración)

```
(deftemplate nota
      (slot altura
            (type SYMBOL)
            (allowed-values A A# B C C# D D# E F F# G G# s)
      )
      (slot octava
            (type INTEGER)
            (allowed-values 0 1 2 3 4 5 6 7 8 9)
      )
      (slot duracion
            (type INTEGER)
            (allowed-values 1 2 3 4 5 6 7 8)
      )
)
```

ARMONÍA MUSICAL

Otra parte del modelo tiene como función representar la **armonía musical** y se compone de las siguientes plantillas:

- **Escala**: Una escala es un conjunto de notas musicales con unos intervalos determinados entre las notas de la propia escala. Para representarla, simplemente, se ha indicado el tipo de escala (mayor, menor, pentatónica), así como las notas que la componen:

```
(deftemplate escala
      (slot tipo
            (type SYMBOL)
      )
      (multislot notas
            (type SYMBOL)
            (allowed-values A A# B C C# D D# E F F# G G#)
      )
)
```

- **Melodía.** Una melodía es una lista de notas

```
(deftemplate melodia
      (multislot notas
            (type SYMBOL)
      )
)
```

- **Intervalos**. Un intervalo representa una transición entre dos notas. El campo estabilidad refleja cómo de deseable es esa transición (estable, semiestable o inestrable)

`(intervalo` *tipo-intervalo nota_origen nota_destino tipo_intervalo*`)`

EJ:
```
(intervalo 2m ?nota ?min2 inestable)
      (intervalo 2M ?nota ?May2 semiestable)
      (intervalo 3m ?nota ?min3 estable)
```

COMPOSICIÓN MUSICAL

Por último ha sido necesario representar los siguientes conceptos que han servido para la lógica de la **composición musical**:

Candidatos: Representa las notas candidatas en un momento determinado de la melodía, y que tienen la misma estructura que una nota musical, ya que los candidatos son notas.

```
(deftemplate candidatos
	(slot altura
		(type SYMBOL)
		(allowed-values A A# B C C# D D# E F F# G G# s)
	)
	(slot octava
		(type INTEGER)
		(allowed-values 0 1 2 3 4 5 6 7 8 9)
	)
	(slot duracion
		(type INTEGER)
		(allowed-values 1 2 3 4 5 6 7 8)
	)
	 (slot tipo
		(type SYMBOL)
		(allowed-values estable semiestable inestable)
	)
)
```

Limitadores: Son valores que limitan a un intervalo determinadas características de la composición. Los limitadores (al contrario de los calibradores que veremos luego) no definen intervalos flexibles, sino que establecen un límite no sobre pasable en ningún caso por las notas de la melodía.

```
(deffacts especificaciones_generador_patrones
	(margenOctavas 2 4)
	(margenDuracionNotas 16 32)
	(margenSilencios 15 25) ;; Probabilidad de silencio / nºnotas
	(margenProgresionCromatica 20 30)
)
```

Estructura de la melodía y patrones: Define la estructura de patrones que seguirá la melodía

```
(deffacts estructura_melodia
	(numeroDeCompases 10)
	(estructuraMelodia (idPatron A B C A+ B C A B C A))
)
```

Los **patrones** están definidos de la siguiente manera:

- Una letra (por ejemplo la A) define un patrón. Si esa letra (la A) se repite a lo largo de la melodía, el patrón se repetirá también.
- Una letra seguida de los símbolos O+ (por ejemplo AO+), indica que si ya existe un patrón con esa letra (A), se insertará el mismo patrón, pero incrementando sus notas en una octava.
- Una letra seguida de los símbolos O- (por ejemplo AO-), indica que si ya existe un patrón con esa letra (A), se insertará el mismo patrón, pero decrementando sus notas en una octava.
- Una letra seguida de los símbolos 3- (por ejemplo A3-), indica que si ya existe un patrón con esa letra (A), se insertará el mismo patrón, pero decrementando sus notas en una tercera.
- Una letra seguida de los símbolos 3+ (por ejemplo A3+), indica que si ya existe un patrón con esa letra (A), se insertará el mismo patrón, pero incrementando sus notas en una tercera.

Por ejemplo, de acuerdo a esta estructura la melodía de la canción Yesterday de "The Beattles", quedaría codificada como (AABCABCA)

Calibradores: Los calibradores son contadores que sirven para determinar la situación de la melodía en un momento determinado y así poder decidir con criterios fundamentados los mejores candidatos para la melodía. Se han representado mediante hechos ordenados que cumplen la siguiente estructura.

```
(Nombre_calibrador Valor)
```

```
(deffacts calibradores
            (calAsc 0)
            (calDesc 0)
            (calOct 0)
            (calDuracion 0)
            (calSilencio 0)
            (calTension 0)
      )
```

4.4. ESTRUCTURA DE APLICACIÓN DE LAS REGLAS

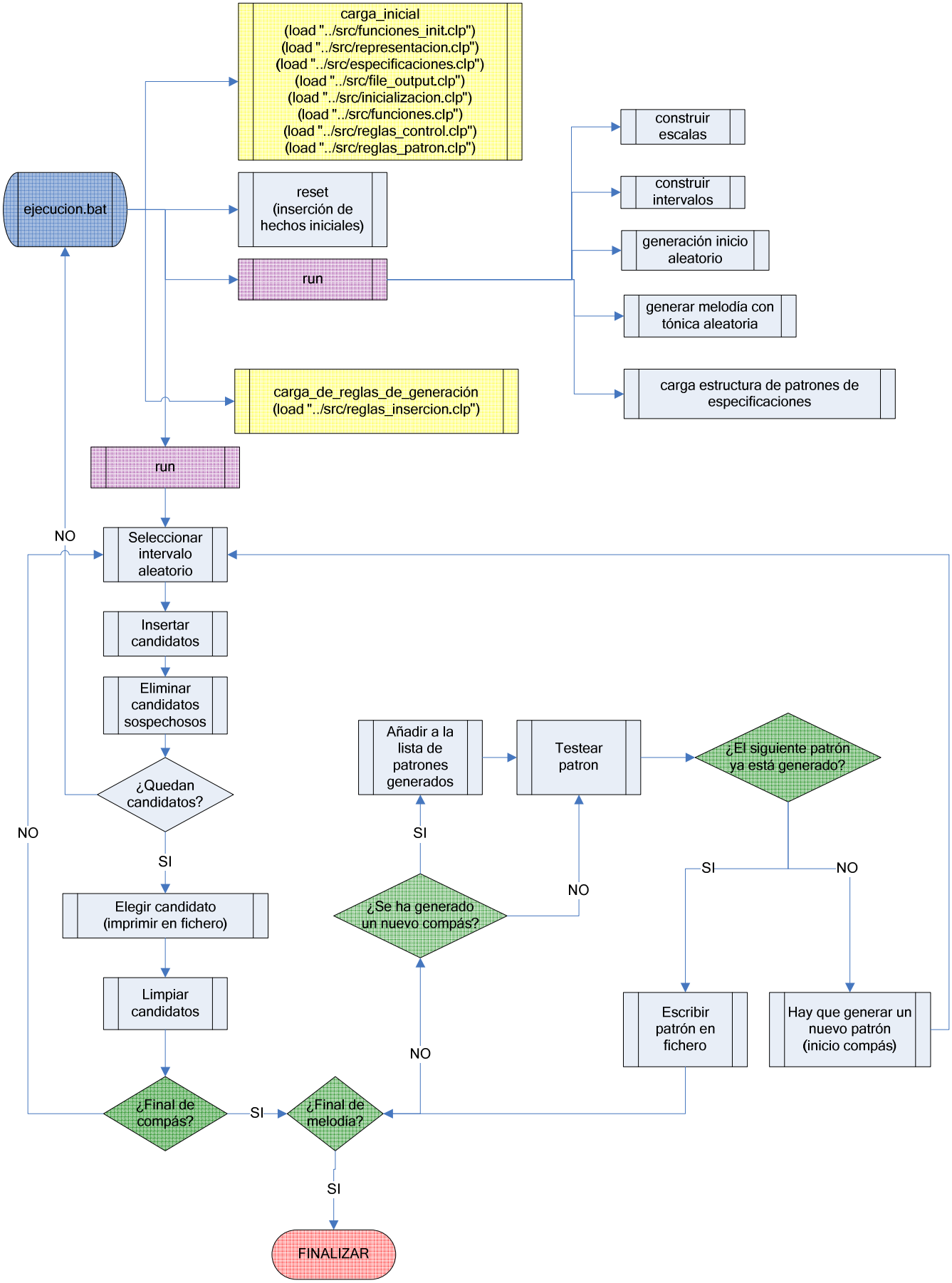


Figura 7 – Diagrama de flujo de ejecución

4.5. DESCRIPCIÓN DE LOS MÓDULOS

Vamos a distinguir dentro del proyecto dos módulos principalmente:

- El módulo del compilador de JAVA (no realizado en este proyecto).
- El módulo de generación de melodía de CLIPS

MÓDULO COMPILADOR DE JAVA

Este módulo no ha sido realizado por este proyecto así pues se dará una escueta información sobre los archivos que lo componen, así como de su funcionamiento básico.

El compilador de Java es un programa que procesa un archivo con extensión .pddl, que contiene una estructura determinada, con la información necesaria para procesar las notas de la melodía.

En el siguiente contenido se muestra el código que se crea después de la generación de cuatro notas (un bloque por nota).

```
(pitch n1 p1)
(octave n1 O4)
(= (speed n1) 100)
(= (length n1) 32)
(= (init n1) 0)
(= (comp n1) 1)

(pitch n1 p7)
(octave n1 O4)
(= (speed n1) 100)
(= (length n1) 16)
(= (init n1) 32)
(= (comp n1) 1)

(pitch n2 p5)
(octave n2 O3)
(= (speed n2) 100)
(= (length n2) 16)
(= (init n2) 48)
(= (comp n2) 1)

(pitch n3 p5)
(octave n3 O4)
(= (speed n3) 100)
(= (length n3) 32)
(= (init n3) 0)
(= (comp n3) 2)
```

En cada una de las líneas que componen un bloque se indica una cualidad de la nota (altura, duración, octava, número de compás, etc.). Cada uno de estos bloques es procesado por el compilador, transformándolo a formato MIDI.

MÓDULO GENERACIÓN DE MELODÍA CLIPS

Este módulo es el principal del proyecto y es el que se encarga de la generación de la melodía musical con el sistema experto CLIPS.

El módulo de CLIPS se subdivide a su vez, en seis módulos principales, que en realidad son seis ficheros que engloban diferentes funcionalidades del sistema general.

1) **Representación del conocimiento** (*representacion.clp*) Este módulo contiene la definición de las plantillas de representación del sistema.

2) **Especificaciones** (*especificaciones.clp)*. Este módulo contiene las especificaciones de entrada al sistema, que son los valores de entrada que requiere el programa para la generación de la melodía en cuestión.

3) **Inicialización del sistema** *(inicializacion.clp).* Este módulo contiene los hechos que inicializan el sistema, tanto las plantillas necesarias, como los contadores, como la generación de los intervalos y escalas.

4) **Reglas de control de la melodía** *(reglas_control.clp).* Este módulo contiene las reglas de control de la melodía. Son las que controlan el estado de la melodía.

5) **Reglas de inserción de notas en la melodía** *(reglas_insercion.clp).* Este módulo contiene las reglas que se encargan de controlar la inserción de las notas que han sido seleccionadas entre los candidatos para su inserción en la melodía.

6) **Funciones** *(funciones.clp).* Este módulo contiene las funciones que son utilizadas por los módulos anteriores para llevar a cabo sus operaciones.

4.6. REGLAS DE GENERACIÓN DE MELODÍAS

En este capítulo iremos recorriendo el diagrama de flujo del punto anterior para ir en ese sentido analizando las reglas que se aplican, por qué se aplican y qué sentido tienen dentro del sistema. Las reglas se estructuran en torno al orden de ejecución y a unos bloques homogéneos que se encuadran de una determinada etapa dentro de la ejecución.

FASE DE INICIALIZACIÓN

Durante esta fase se realizan las siguientes funciones:

- Creación de todas las estructuras necesarias para la generación de la melodía.
- Creación de duración y octava aleatorias iniciales.
- Recogida de las especificaciones del usuario.

- **construirIntervalos**
  Inserta en la base de hechos los intervalos que luego servirán para construir las escalas, asignándole directamente a cada uno un carácter (estable | semiestable o estable) según la estabilidad que ofrece el intervalo a la melodía.

- **construirEscalas**
  Una vez generados los intervalos, genera las escalas disponibles para la creación de la melodía. En nuestro caso se genera la escala mayor y la menor, aunque se podrían añadir todas las que se desearan.

- **construirApoyoIntervalosSilencios**
  Para cada uno de los posibles intervalos generados, se añade la posibilidad de un "intervalo a silencio" que no es más que un intervalo desde un silencio a cualquier nota de la escala.

- **duracionAleatoria**
  Genera duraciones aleatorias hasta que se encuentre una duración aleatoria que se encuentre dentro de los márgenes establecidos en las especificaciones.

- **comprobarDuracionAleatoria**
  En el antecedente equipara los márgenes de duración de las especificaciones y la duración aleatoria generada por la regla duracionAleatoria y llama a la función comprobarDuracionAleatoria que informa (insertando hechos) si la duración aleatoria es permitida o no.

- **octavaAleatoria**
  Genera octavas aleatorias hasta que se encuentre una octava aleatoria que se encuentre dentro de los márgenes establecidos en las especificaciones.

- **comprobarOctavaAleatoria**
  En el antecedente equipara los márgenes de octavas de las especificaciones y la octava aleatoria generada por la regla octavaAleatoria y llama a la función comprobarDuracionAleatoria que informa (insertando hechos) si la octavaaleatoria es permitida o no.

- **insertarTonica**
  Inserta definitivamente la tónica para dar paso al proceso de composición, una vez que en

el antecedente se encuentra una duración aleatoria comprobada, una octava aleatoria comprobada, la escala, el modo y la tónica. Estos tres últimos parte de las especificaciones de la melodía.

FASE DE CONTROL A NIVEL DE NOTA

- **`seleccionarIntervalo`**

  Equipara en su antecedente un intervalo de manera aleatoria y llama a la función *insertarCandidatos*, que genera un conjunto de hechos con los posibles candidatos como siguiente nota a seleccionar en la melodía.

- **`eliminarCandidatosSospechosos`**

  Recoge los límites fijados en las especificaciones así como los calibradores y llama la función `comprobarCandidato` que filtra los candidatos de dos maneras:

  - Si se sobrepasa el límite de los márgenes se eliminan
  - Según el valor de los calibradores serán eliminados o no de manera probabilística de tal manera que cuanto más alto (o cercano al límite) se encuentre el calibrador, más posibilidades existen de ser eliminado.

- **`elegirCandidato`**

  Elige uno de los candidatos que han pasado el filtro y lo inserta en la melodía. Una vez insertada, llama a la función `imprimirNotaEnFichero` y `actualizarCalibradores`, que imprime la nota recién generada al fichero de melodía y actualiza el valor de los calibradores según el candidato seleccionado respectivamente.

  También genera un hecho `(candidato elegido)` que es el que procede a la limpieza de candidatos de la base de hechos y el que desencadena el proceso de generación de datos y elección de nota a insertar.

- **`limpiarCandidatos`**

  Limpia la base de hechos de candidatos.

- **`finLimpiarCandidatos`**

  Detecta el momento en que se han limpiado los candidatos, insertado la nota y la base de hechos está preparada para empezar la nueva generación de candidatos.

FASE DE CONTROL A NIVEL DE COMPÁS (PATRONES)

- **finCompás**

  Detecta el final del compás a partir del contador de posición en compás. Una vez finalizado el compás, se pueden dar 2 situaciones:

  - Que se haya llegado al número total de compases, entonces se finaliza la ejecución.
  - Que todavía queden compases, con lo que se inserta un hecho (assert (testearPatron)) que hace que se active la regla que lleva el mismo nombre que el hecho.

- **testearPatron**

  Una vez se ha finalizado el compás, esta función analiza la estructura de la melodía realizando, según el siguiente patrón a generar, las siguientes funciones:

  - Si el siguiente patrón ya existe en la lista de patrones generados, marca la base de hechos para que se imprima el patrón ya existente.
  - Si el siguiente patrón no existe en la lista de patrones generados, marca la base de hechos para que se vuelva a realizar todo el proceso de generación de notas y patrones.

  También se extrae el tipo de patrón, de tal manera que si viene primero la nota traducida (de la A a la L) los dos siguientes caracteres serán la modificación de los patrones que puede ser:

  - O+ patrón una octava por encima
  - O- patrón una octava por debajo
  - 3+ patrón una tercera por encima
  - 3- patrón una tercera por debajo

- **imprimirPatron**

  Imprime un patrón ya generado o bien un patrón ya generado modificado.

  Se coge el tipo de patrón recibido (del que informa testarPatron) y a partir de ese tipo va imprimiendo el patrón en el fichero.

4.7. FUNCIONES

Vamos a ver en primer lugar las funciones que se encuentran en funciones_init.clp

- **`comprobarDuracionAleatoria`**

  Esta función recibe cuatro parámetros, los dos primeros que pertenecen a la duración aleatoria generada y los dos siguientes, que son los márgenes definidos en la inicialización que no se pueden sobrepasar. Según si la duración aleatoria generada

está dentro de los límites inserta un hecho positivo o negativo.

- **comprobarOctavaAleatoria**
  Esta función recibe unos parámetros con la duración aleatoria generada. Según si la octava aleatoria generada está dentro de los límites inserta un hecho positivo o negativo.

- **traducirNotaM**
  Traduce una nota que recibe (C, C#, B) en una letra equivalente que ocupe solo un carácter (es necesario para que las notas de la melodía ocupen el mismo número de caracteres):
  (C → A, C# → B, D → C, D# → D, ...)

- **traducirNotaInv**
  Hace la traducción inversa de la función anterior.

- **traducirDuracion** (?d)
  Traduce la duración para que ocupe un único carácter (es necesario para que las notas de la melodía ocupen el mismo número de caracteres):
  (1 → "1", 2 → "2", 4 → "3", 8 → "4", 16 → "5", …)

- **traducirDuracionInv** (?d)
  Hace la traducción inversa de la función anterior.

- **traducirDuracionInvNum** (?d)
  Traduce la salida de **traducirDuracion** en símbolo, ya que con las cadenas de caracteres no se puede trabajar.

Vamos a ver ahora las funciones que se encuentran en funciones.clp

- **mismaOctava**
  Devuelve 1 si entre dos notas se ha producido un salto de escala (si la diferencia entre la posición de ambas notas en la escala cromática es negativa o positiva) y 0 si no se ha producido.

- **ascendente**
  Devuelve 0 si entre la nota actual y la nota generada se ha producido un salto descendente en altura, 1 se ha sido ascendente y 2 si no se ha producido un cambio de altura entre las dos notas a comparar.

- **comprobarValidezInsercion**

- **`comprobarCandidato`**

  Esta función recibe un candidato y se filtra según:

  - Los calibradores, de tal manera que se genera un intervalo aleatorio para cada calibrador y si el valor del candidato supera ese intervalo aleatorio generado, el candidato será eliminado.
  - Los márgenes, de tal manera que si se superan los márgenes (de las especificaciones iniciales) se eliminará el candidato.

- **`rellenarConSilencio`**

  Esta función inserta un silencio en la melodía.

- **`insertarCandidatos`**

  Genera hechos candidatos que varían en altura (octavas) y duración. Dispone de un filtro que limita la posibilidad de que existan candidatos que se "salten" un compás o que se salgan del límite de octavas existente en las especificaciones.

  Una vez insertados se genera un hecho `(assert (eliminando sospechosos))` que llama al filtrado de estos candidatos según los calibradores.

- **`actualizarCalibradores`**

  Se encarga de actualizar el valor de todos los calibradores de la base de hechos. Para explicar su funcionamiento, se analizará los calibradores de altura (ascendente y descendente). Se pueden producir varias situaciones:

  - Si la nota crece en altura
    - El calibrador de altura ascendente se incrementa en 1
    - El calibrador "opuesto" (calibrador de altura descendente se pone a 0)
  - Si la nota decrece en altura
    - El calibrador de altura descendente se incrementa en 1
    - El calibrador "opuesto" (calibrador de altura ascendente se pone a 0)
  - Si la nota no varía en altura
    - Se ponen ambos calibradores a -1, para que no haya más de 3 notas seguidas de la misma altura.

  Este proceso se realiza con cada uno de los calibradores, que según su formato, tendrán comportamientos distintos.

- **`terceraPorEncimaDe`**

  Recibe una nota y devuelve otra que será la tercera por encima de la recibida.

5. EVALUACIÓN

En este punto se muestran los resultados que se han obtenido en las diferentes pruebas de generación de la melodía, se evalúa el alcance que se ha conseguido con el proyecto y se aportan unas conclusiones sobre lo experimentado.

5.1. DISEÑO DE EXPERIMENTOS

En este apartado vamos a exponer los experimentos que se realizarán sobre el sistema para evaluar los resultados que arroja. Para ello realizaremos tres iteraciones sobre cuatro pruebas, cada una de ellas con una configuración diferente.

1) **Generación de notas aleatorias**. En esta prueba se generarán notas aleatoriamente sin tener en cuenta ninguna regla armónica o musical.
2) **Generación de notas aleatorias dentro de una escala predefinida**. Se generarán notas aleatorias dentro de un grupo de notas pertenecientes a una escala musical concreta. No se tendrá en cuenta regla armónica o musical alguna.
3) Generación de notas aleatorias dentro de una escala aleatoria y además estableciendo una limitación a partir de una configuración inicial y de unos **calibradores** que establecerán márgenes "lógicos" dentro de la composición.
4) Además de todo lo anterior se incluirá la **repetición de patrones**, de tal manera que las melodías se construirán en base a patrones repetitivos o modificados.

5.2. RESULTADOS EXPERIMENTALES

PRUEBA 1-1

Duración: 16 compases.

Escala: cromática

MargenOctavas: 1- 8

MargenDuracionNotas: 1 64 (de semifusa a redonda)

MargenSilencios 1 15

RESULTADOS PRUEBA 1-1

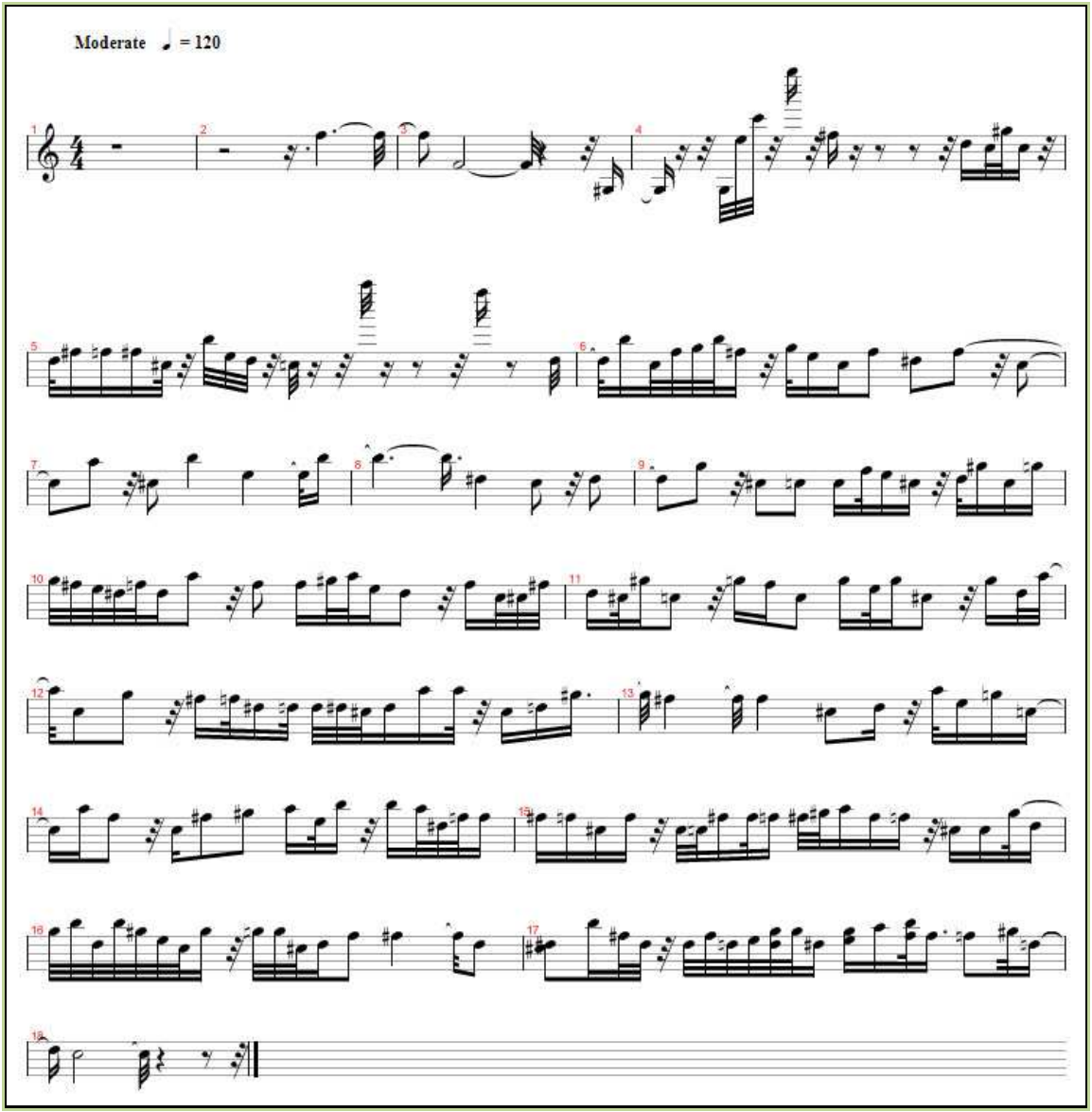


Ilustración 8 – Prueba 1-1

PRUEBA 1-2

Duración: 16 compases.

Escala: cromática

MargenOctavas: 1- 8

MargenDuracionNotas: 1 64 (de semifusa a redonda)

MargenSilencios 1 15

Estructura: * * * * * * * * * * * * * * * *

RESULTADOS PRUEBA 1-2

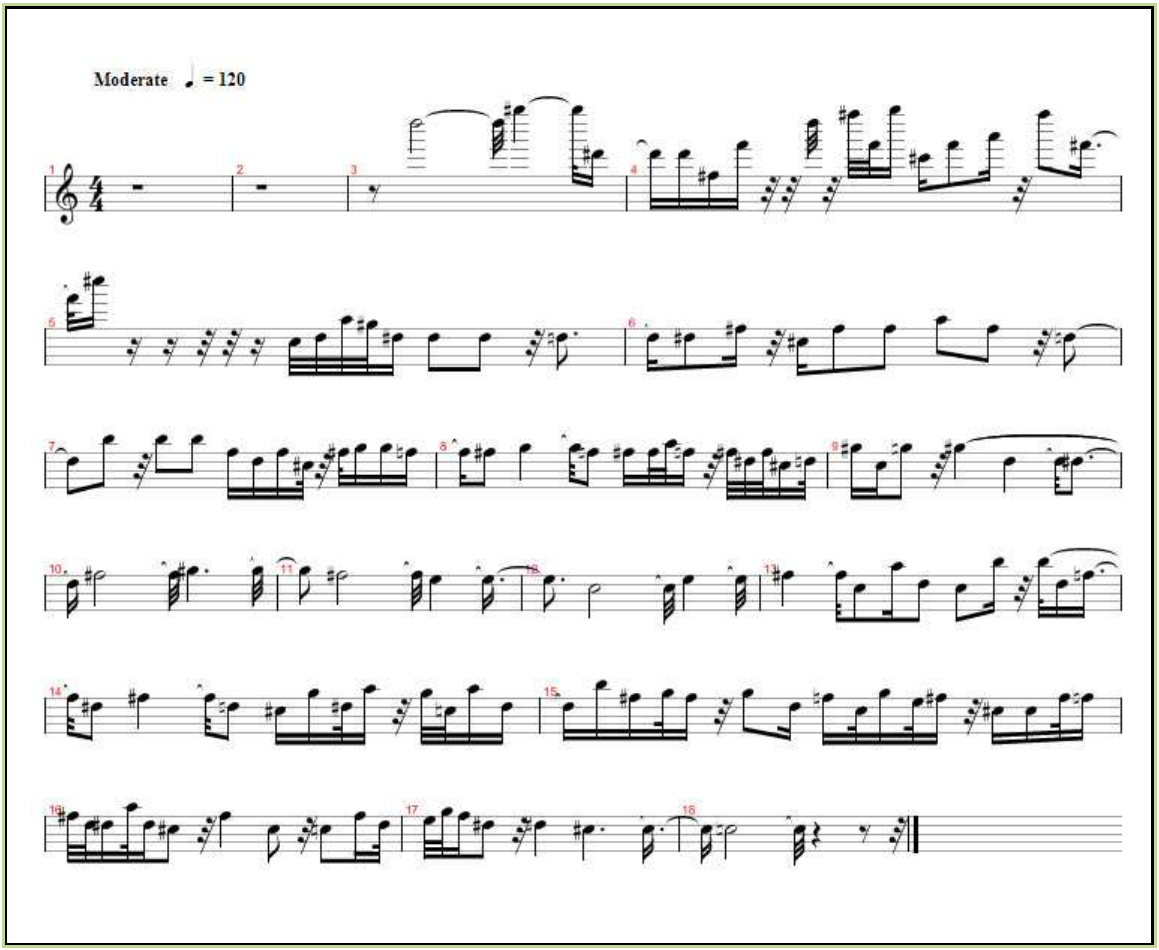


Ilustración 9 – Prueba 1-2

PRUEBA 1-3

Duración: 10 compases.

Escala: cromática

MargenOctavas: 1- 8

MargenDuracionNotas: 1 64 (de semifusa a redonda)

MargenSilencios 1 15

Estructura: * * * * * * * * * * * * * * * *

RESULTADOS PRUEBA 1-3

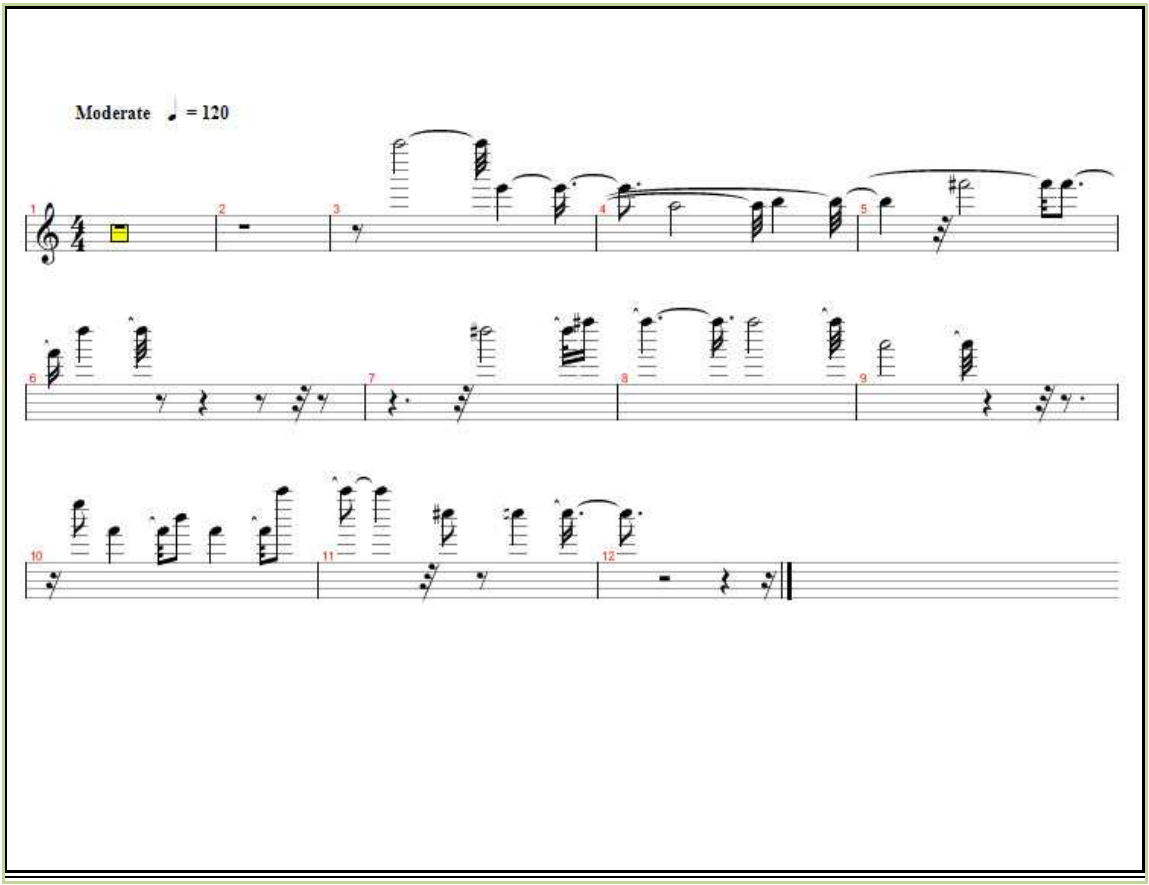


Ilustración 10 – Prueba 1-3

PRUEBA 2-1

Duración: 16 compases.

Escala: cromática

MargenOctavas: 1- 8

MargenDuracionNotas: 1 64 (de semifusa a redonda)

MargenSilencios 1 15

Tónica: C

Modo: Mayor

Estructura: * * * * * * * * * * * * * * * *

RESULTADOS PRUEBA 2-1

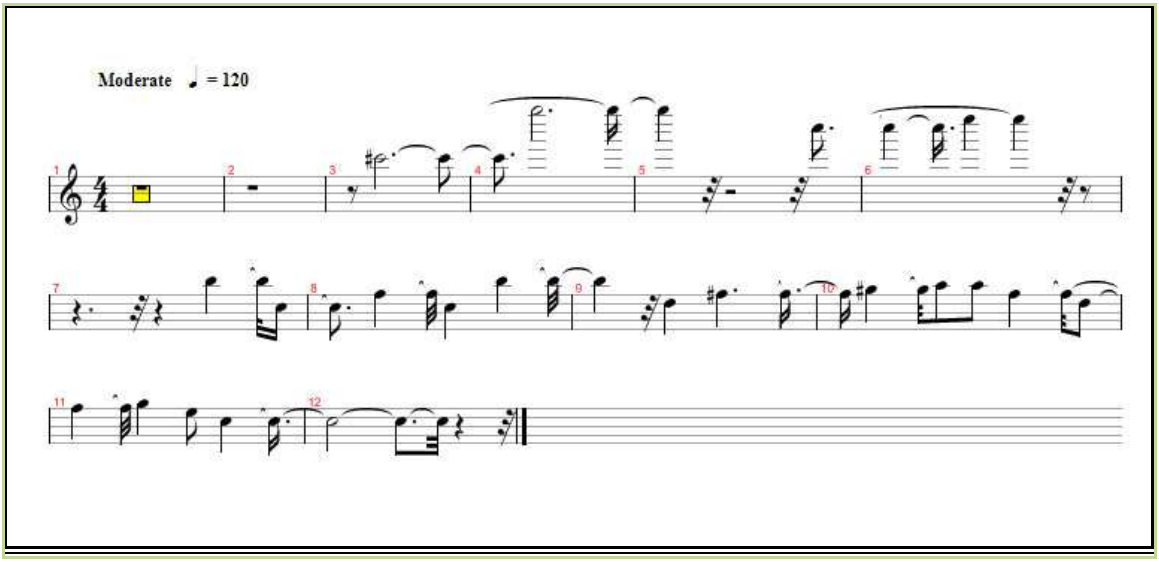


Ilustración 11 – Prueba 2-1

PRUEBA 2-2

Duración: 16 compases.

Escala: cromática

MargenOctavas: 1- 8

MargenDuracionNotas: 1 64 (de semifusa a redonda)

MargenSilencios 1 15

Tónica: D

Modo: menor

Estructura: * * * * * * * * * * * * * * * *

RESULTADOS PRUEBA 2-2

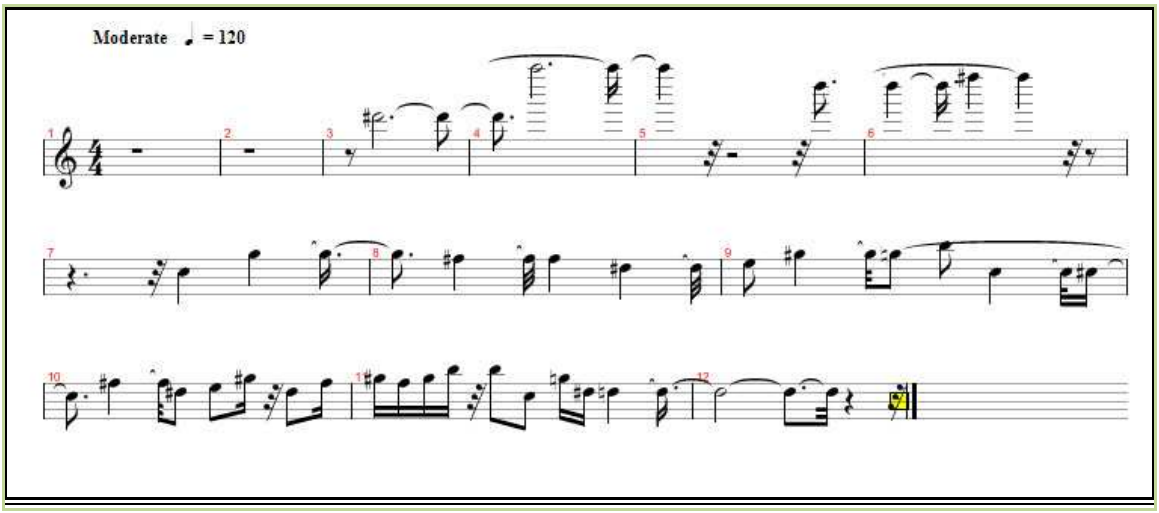


Ilustración 12 – Prueba 2-2

PRUEBA 2-3

Duración: 16 compases.

Escala: cromática

MargenOctavas: 1- 8

MargenDuracionNotas: 1 64 (de semifusa a redonda)

MargenSilencios 1 15

Tónica: F

Modo: mayor

Estructura: * * * * * * * * * * * * * * * *

RESULTADOS PRUEBA 2-3

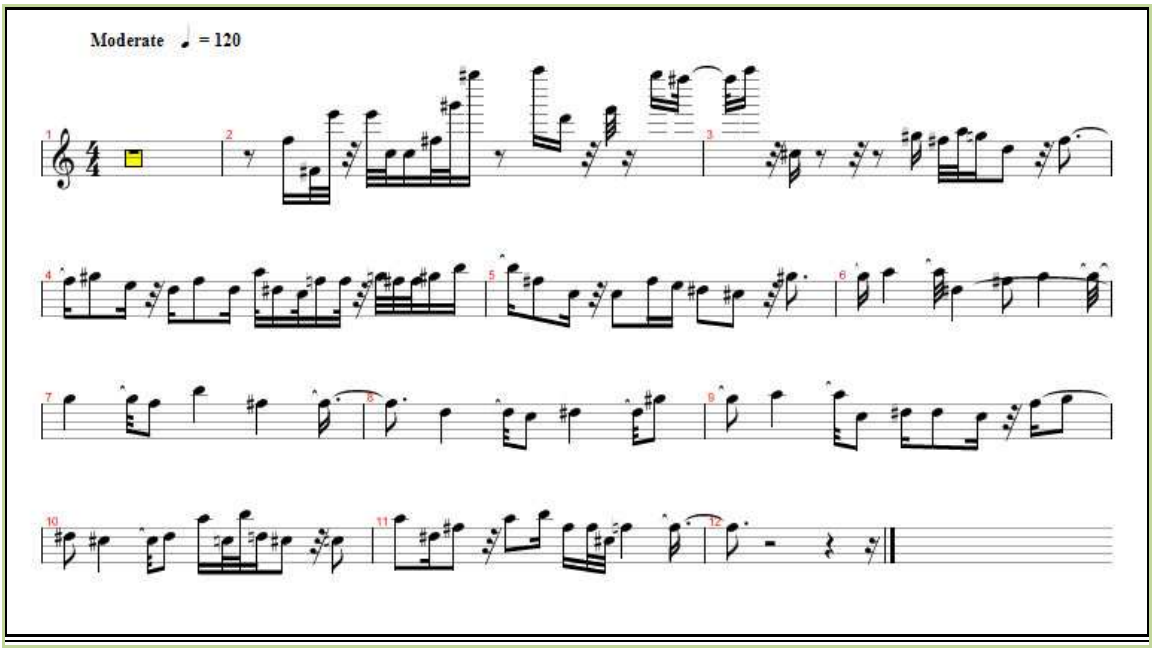


Ilustración 13 – Prueba 2-3

PRUEBA 3-1 (MUY RESTRICTIVA)

```
(deffacts especificaciones_compilador
        (tonica C)
        (modo menor)
        (timbre 8)
        (pulso 60)
        (armadura
                (duracion 4)
                (numero 2)
        )
)
 (deffacts especificaciones_generador_patrones
        (margenOctavas 3 3)
        (margenDuracionNotas 16 16)
        (margenSilencios 20 25) ;; especifica el margen de probabilidad de silencio / nºnotas
)
```

RESULTADOS PRUEBA 3-1

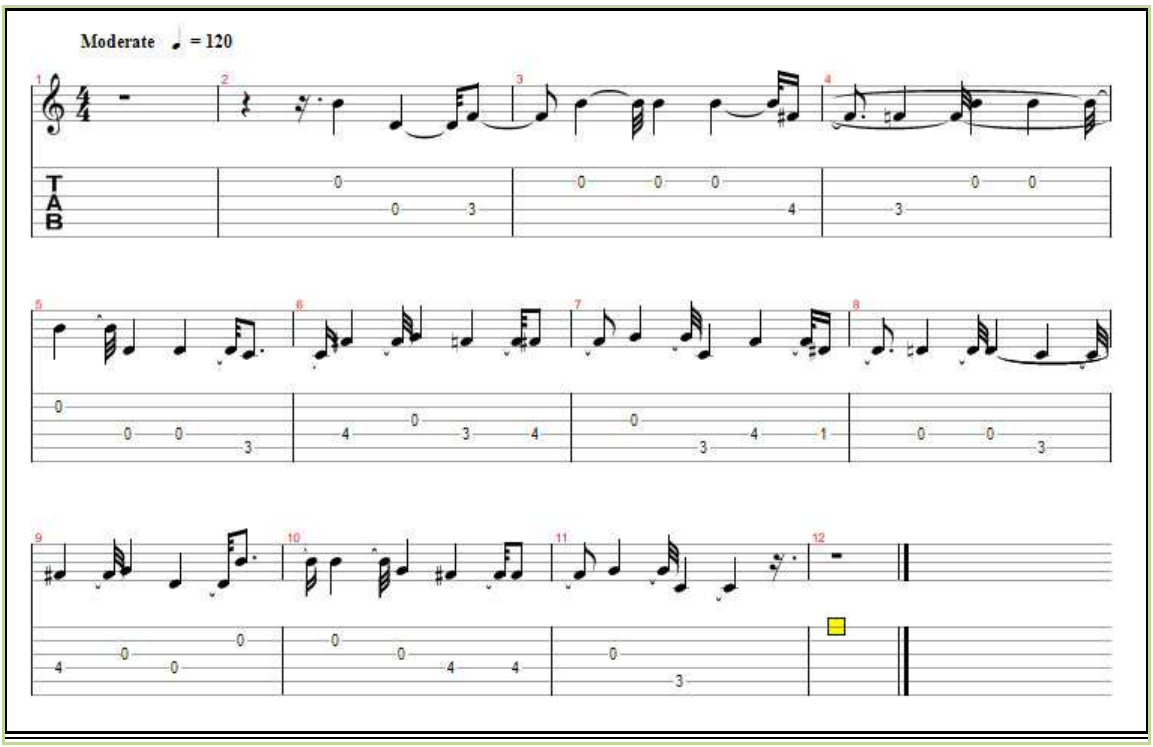


Ilustración 14 – Prueba 3-1

PRUEBA 3-2 (NORMAL)

(deffacts especificaciones_compilador
(tonica C)
(modo menor)
(timbre 8)
(pulso 60)
)
(deffacts especificaciones_generador_patrones
(margenOctavas 3 4)
(margenDuracionNotas 8 32)
(margenSilencios 20 25) ;; especifica el margen de probabilidad de silencio / nºnotas
)

RESULTADOS PRUEBA 3-2

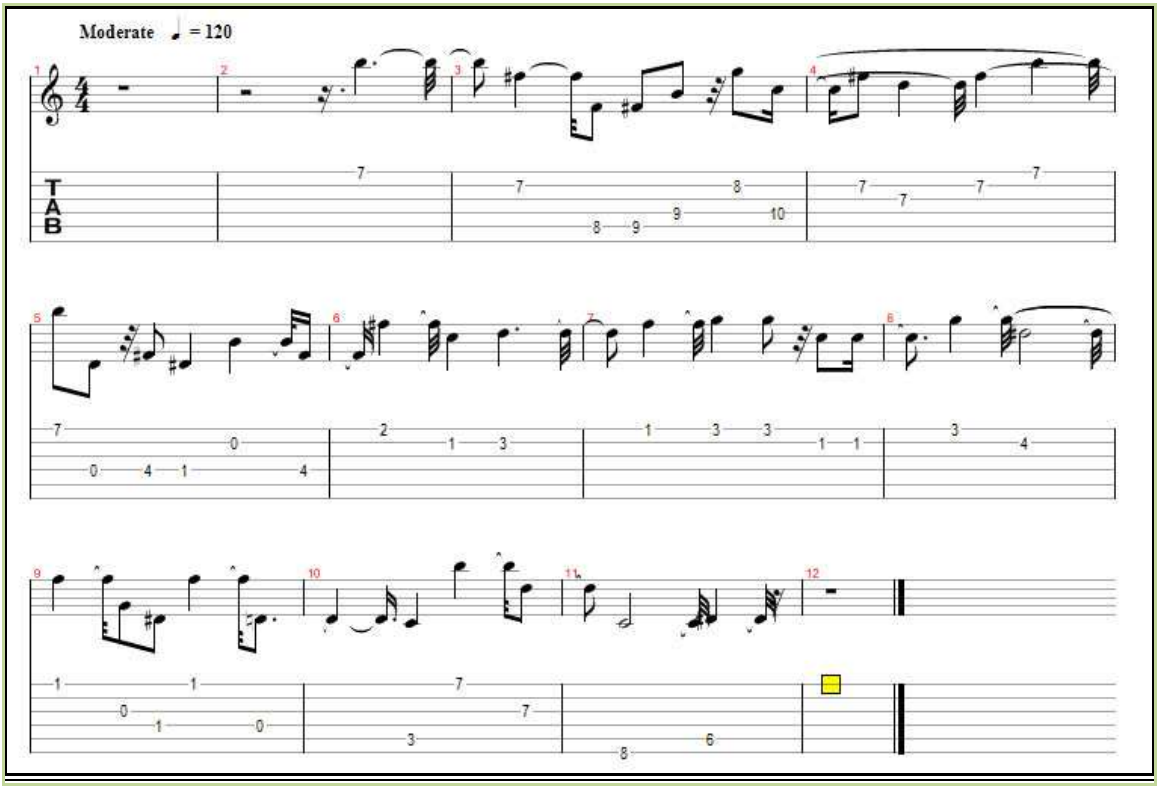


Ilustración 15 – Prueba 3-2

PRUEBA 3-3 (POCO RESTRICTIVA – PRUEBA DE CALIBRADORES SECUNDARIOS)

(deffacts especificaciones_compilador

(tonica C)

(modo menor)

(timbre 8)

(pulso 60)

)

(deffacts especificaciones_generador_patrones

(margenOctavas 1 8)

(margenDuracionNotas 1 64)

(margenSilencios 20 25) ;; especifica el margen de probabilidad de silencio / nºnotas

)

RESULTADOS PRUEBA 3-3

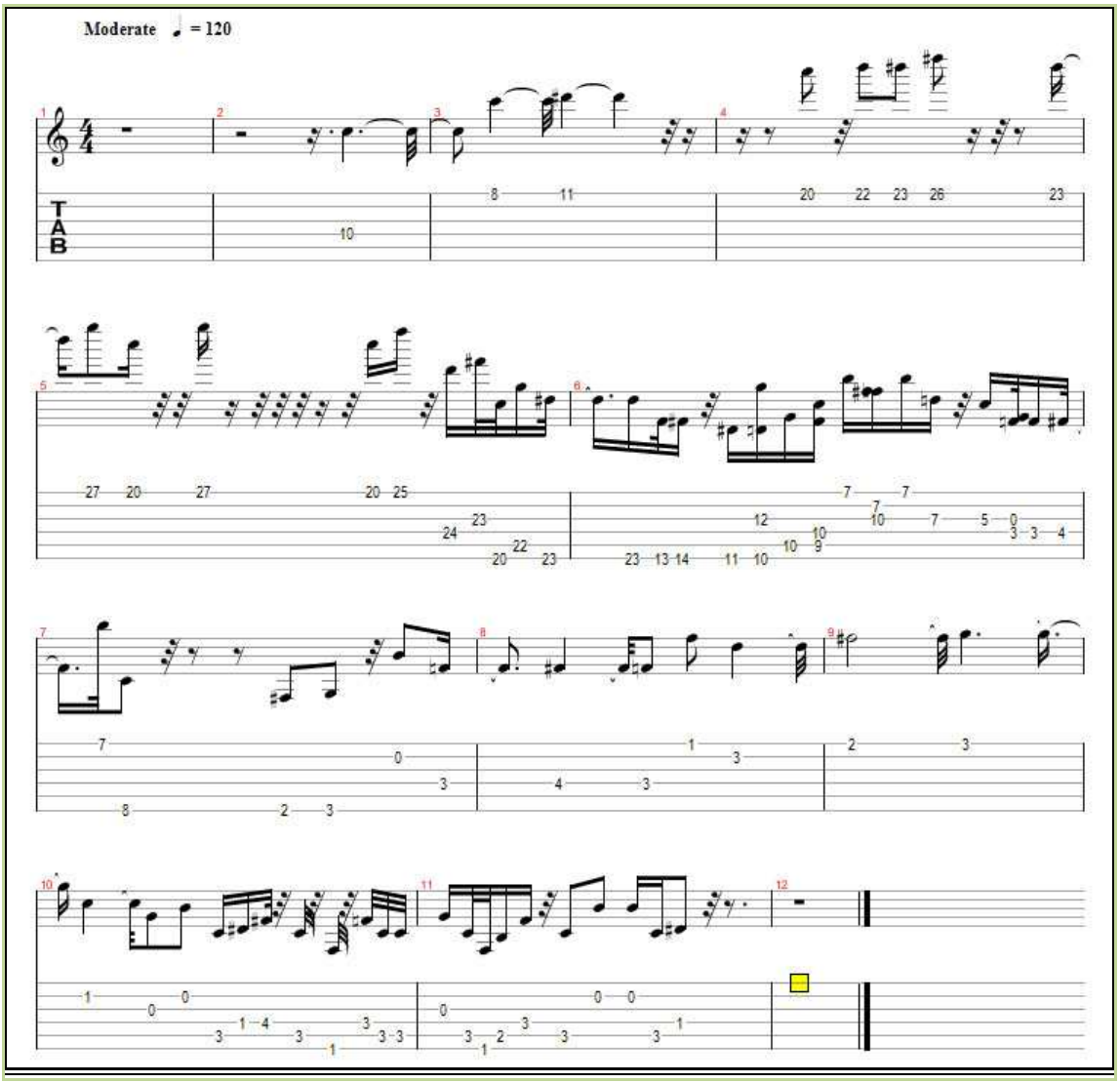


Ilustración 16 – Prueba 3-3

PRUEBA 4-1

```
(deffacts especificaciones_generador_patrones
        (margenOctavas 3 4)
        (margenDuracionNotas 8 32)
        (margenSilencios 20 25)
)
(deffacts estructura_melodia
        (numeroDeCompases 9)
        (estructuraMelodia (idPatron A B C A B C A B C))
)
```

RESULTADOS PRUEBA 4-1

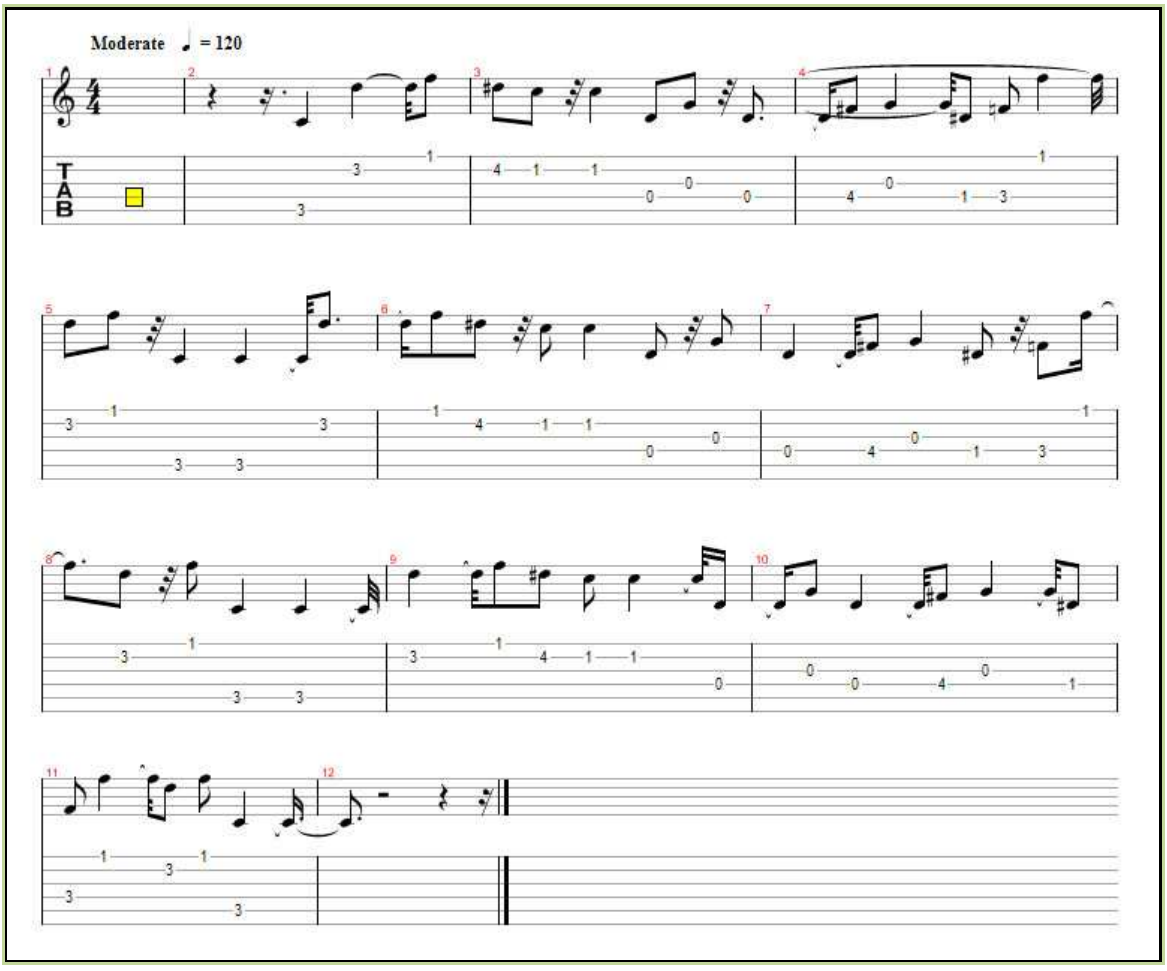


Ilustración 17 – Prueba 4-1

PRUEBA 4-2

```
(deffacts especificaciones_generador_patrones
        (margenOctavas 3 4)
        (margenDuracionNotas 16 32)
        (margenSilencios 20 25)
)

(deffacts estructura_melodia
        (numeroDeCompases 12)
        (estructuraMelodia (idPatron A B A B C D A B A B C D))
)
```

RESULTADOS PRUEBA 4-2

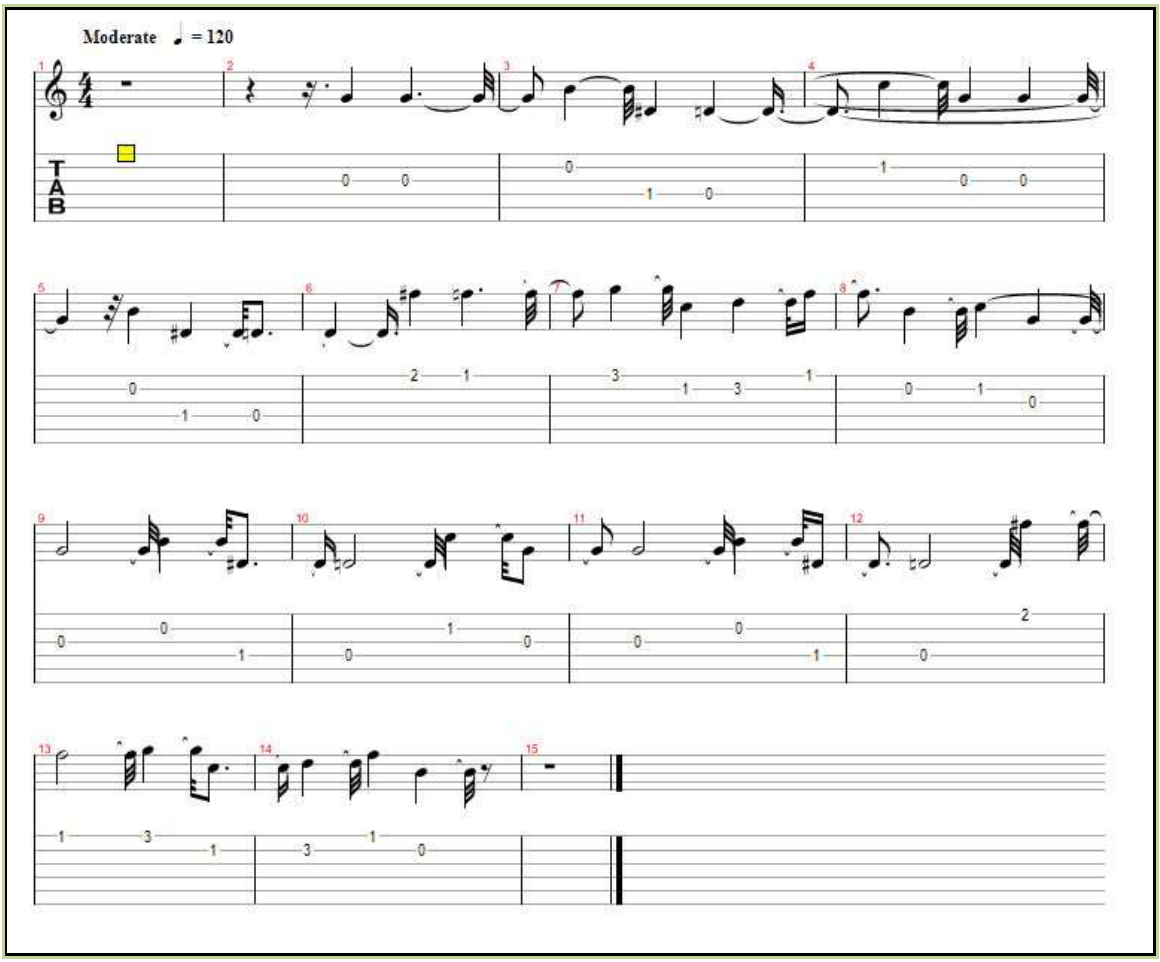


Ilustración 18 – Prueba 4-2

PRUEBA 4-3

```
(deffacts especificaciones_generador_patrones
        (margenOctavas 3 4)
        (margenDuracionNotas 16 32)
        (margenSilencios 20 25)
)
(deffacts estructura_melodia
        (numeroDeCompases 12)
        (estructuraMelodia (idPatron A B C D AO+ BO+ C D A B CO+ DO+))
)
```

RESULTADOS PRUEBA 4-3 (EN EL ANEXO D SE PUEDE VER UN EJEMPLO DE LA SALIDA POR PANTALLA DE LA EJECUCIÓN DE ESTA PRUEBA)

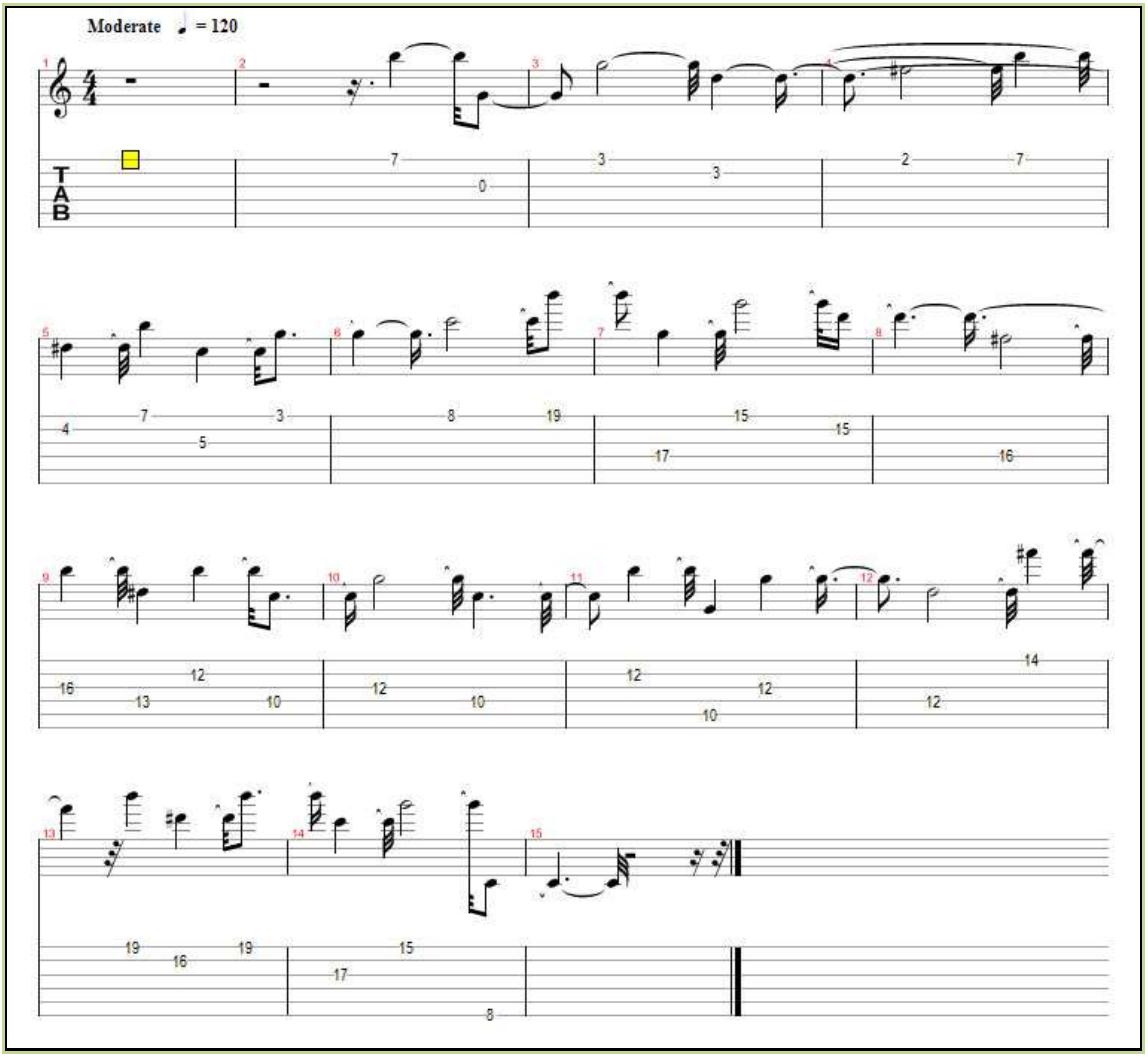


Ilustración 19 – Prueba 4-3

5.3. VALORACIÓN DE RESULTADOS

Las pruebas se han dividido en cuatro partes

PRUEBA 1 (SIN NINGUNA RESTRICCIÓN)

En las tres pruebas pertenecientes a esta parte, se observa una absoluta anarquía en la generación de notas:

- Numerosos silencios seguidos.
- Dos notas seguidas que difieren tres octavas en altura.
- Una nota blanca que transita a una nota semifusa.
- Ninguna estructura ni limitación.

PRUEBA 2 (RESTRICCIÓN DE ESCALA)

Poco se puede observar. La única observación con respecto a las pruebas anteriores es que, efectivamente, las notas seleccionadas pertenecen a una escala determinada.

PRUEBA 3 (RESTRICCIÓN DE CALIBRACIÓN)

En este caso ya se pueden observar corregidos todos los problemas que nos encontrábamos en las anteriores pruebas, ya que se han activado los calibradores de:

- **Altura**, que, como se puede observar en las partituras, actúan para que dos notas no disten demasiado en altura.
- **Duración**, que impide que una nota blanca sea seguida de una semifusa, como veíamos en las primeras pruebas y produce una melodía de duración más acordes a la composición humana.
- **Silencios**, que impide que existan muchos silencios seguidos, creando unos márgenes lógicos de inserción de silencios.
- **Entorno**, que controla que no pueda haber una sucesión indeterminada de notas ascendentes o descendentes indeterminada, sino que, como las reglas de la armonía dictan, cuanto más ascienda una melodía más tiende a descender y viceversa.

PRUEBA 4 (INTRODUCCIÓN DE PATRONES)

En el último caso se puede observar la repetición de patrones y la homogeneidad que adquiere la estructura de las partituras al repetirse estructuras o modificarse respecto a las octavas y terceras.

6. CONCLUSIONES

A partir del trabajo realizado y de los objetivos que se han alcanzado, se pueden extraer del trabajo las siguientes conclusiones:

- Es posible, mediante la abstracción, extraer los métodos fundamentales que utiliza un compositor al componer una obra y, teniendo en cuenta estos aspectos, se puede generar una melodía que simule la que podría realizar un compositor a partir de una escala musical.

- Lo que le da homogeneidad a una composición musical, es el reconocimiento de diferentes patrones a lo largo de la canción, de tal manera que al escuchar un patrón repetido resulta una sensación de estabilidad.

- El sistema CLIPS resulta una buena herramienta a la hora de realizar este sistema experto, sin embargo, el hecho de que no posea funciones probabilísticas, limita la capacidad del presente proyecto de gestionar mejor las posibilidades de transición entre notas.

7. LÍNEAS FUTURAS

Las líneas futuras que se presentan en este punto hacen referencia a las posibles aplicaciones futuras que pueda tener la tecnología utilizada en este proyecto y la utilidad que se le puede dar para fines aplicados.

Sin duda una de las carencias de este tipo de sistemas es que no han sido aplicables de manera "funcional" al quehacer diario de músicos y compositores. Creo que es el momento de madurez perfecto para que este tipo de sistemas empiecen a enfocarse a tareas que realmente sean aplicables a una función "social". Dentro de las posibles aplicaciones futuras de este tipo de sistemas voy a destacar cuatro:

1) **Psicología musical (La canción perfecta)**: Muy conocida es la frase "La música amansa a las fieras" ¿y si se tuviera una completa experiencia sobre las emociones o estímulos que provoca la música sobre las personas? Entonces se podrían elaborar composiciones musicales que bien podrían servir para subsanar determinados estados mentales o bien para transportarnos a estados de animo concretos. Si un ordenador pudiera llegar a entender todo esto, seguramente sería capaz de hacer una composición que nos hiciera llorar o que nos hiciera olvidarnos de todas las preocupaciones.

   Sin duda el mayor escollo que tendría que superar este sistema en potencia, sería el de lograr encontrar los patrones musicales que generan determinadas percepciones sensoriales.

2) **Didactica musical**. La didáctica musical podría ser uno de los grandes beneficiados de este tipo de sistemas. Sistemas que fueran capaces de enseñar música a personas sobre todo iniciadas, que pudieran determinar la calidad de una obra compuesta por alguien, o los errores que tiene en su composición.

   Este sistema podría reconocer lo que está interpretando el usuario y ofrecer una evaluación completa tanto de la calidad de la composición, como de la calidad de la interpretación, así como de posibles errores que se hayan producido o mejoras que se pudieran incorporar.

3) **Acompañamiento en directo**. Si un sistema fuera capaz de "escuchar una canción" y de improvisar sobre ella, sería también capaz de acompañar a cualquier músico en directo, incluso sin tener total conocimiento de lo que se está tocando, analizando y procesando en tiempo real lo que está "escuchando" y pudiendo acompañar una música que estuviera interpretándose en directo.

4) **Resolución del bloqueo del compositor**. Otra posible aplicación de este tipo de sistemas sería la de apoyo al compositor. Lo que he llamado "bloqueo del compositor" es una situación en la que el compositor tiene una melodía y o bien no le acaba de convencer, o bien no sabe como continuar. Podría elaborarse un sistema que fuera capaz de recomendar al compositor qué camino puede seguir, sugerirle la utilización de un determinado acordes, o sugerir, a partir de la armonía introducida, qué escala sería la más adecuada para improvisar encima.

8. BIBLIOGRAFÍA Y RECURSOS ELECTRÓNICOS

8.1. BIBLIOGRAFÍA

Barucha, J. (1993). A Connectionist Model of Musical Harmony. . *In Machine Models of Music* (págs. 497-509). Cambridge: M. Schwanauer y D.A. Levitt.

Biles, J. (1994). A Genetic Algorithm for Generating Jazz Solos. *International Computer Music Conference* (págs. 131-137). San Francisco (California): Internation Computer Music Association.

Bresin, R. (2002). Articulation Rules for Automatic Music Performance. *International Computer Music Association*, (págs. 239-270).

Calvo Cuenca, A. (2005). *Programación en lenguaje CLIPS.* Centro de Estudios Ramón Areces.

Canazza, S. (1997). Artificial Neural Networks-Based Models for Automatic Performance in a Clarinet Performance. *International Computer Music Association*, (págs. 113-120). San Francisco (California).

Cope, D. (1987). Experiments in Music Intelligence. *International Computer Music Conference* (págs. 174-181). San Francisco (California): International Computer Music Association.

Ebcioglu, K. (1993). An Expert System for Harmonizing Four-Part Chorales. *In Machine Models of Music* (págs. 385-401). Cambridge: S.M. Schwanauer y D.A. Levitt.

Feulner, J. (1993). Neural Networks That Learn and Reproduce Various Styles of Harmonization. *Proceedings of the 1993 International Computer Music Conference*, (págs. 236-239). San Francisco (California).

Fry, C. (1993). A Language for Specifying Musical Style. *In Machine Models of Music* (págs. 427-451). Cambridge: MIT Press.

Isaacson, H. y. (1993). Musical Composition with a High-Speed Digital Computer. *In Machine Models of music.* (págs. 9-21). Cambridge: S.M. Schwanauer y D.A. Levitt.

Johnson, M. (1992). An Expert System for Articulation of Bach Fugue Melodies. *Computer-Generated Music* (págs. 41-51). Washington, D.C.: D.L. Baggi.

Levitt, D. (1993). A Representation for Musical Dialects. *In Machine Models of Music* (págs. 455-469). Cambridge: S.M. Schwanauer y D.A. Levitt.

Moorer, J. (1993). Music and Computer Composition. *In Machine Models of Music* (págs. 167-186). Cambridge: S.M. Schwanauer y D.A. Levitt.

Morales - Manzanares, R. (2001). An Interactive Music Composition System Using Body Movements. *Journal of New Musci Research* , 25-36.

Papadopoulos, G. y. (1998). Genetic Algorithm for the Generation of Jazz Melodies. *Finnish Conference on Artificial Intelligence.* Jyvaskyla (Finlandia).

Penfold, R. (1996). *MIDI: Proyectos prácticos de música electrónica.* Paraninfo.

Reguera Reguera, M. Á. (2000). *Marcas de agua y ficheros MIDI.* M.A. Reguera.

Rumsey, F. (2004). *Desktop audio technology: digital audio and MIDI principles.* Focal Press.

Sabater, J., & Arcos, J. y. (1998). Using Rules to Support Case-Based Reasoning for Harmonizing Melodies. *AAAI Spring Symposium on Multimodal Reasoning.* Standford (California).

8.2. RECURSOS ELECTRÓNICOS

Audiovisual, C. (15 de 02 de 2008). *CSS Audiovisual.* Obtenido de CSS Audiovisual - ¿Qué es el MIDI?
http://www.css-audiovisual.com/areas/guias/midi.htm

CLIPS. (12 de 06 de 2008). *CLIPS Sitio Oficial.* Obtenido de CLIPS Sitio Oficial: http://www.ghg.net/clips/CLIPS.html

Desconocido. (12 de 03 de 2008). *The Linux MIDI How To.* Obtenido de The Linux MIDI How To
http://tldp.org/HOWTO/MIDI-HOWTO-8.html

Hecquet, A. (1990). *Entorno MIDI y sus aplicaciones musicales.* RA-MA.
Instruments, H. (12 de 04 de 2008). *Hinton Instruments.* Obtenido de Hinton Instruments: http://hinton-instruments.co.uk/reference/midi/protocol/index.htm

lab, T. (19 de 06 de 2008). *MIDI how to.* Obtenido de MIDI how to: http://www.tweakheadz.com/how_to_get_started_with_midi.html

Lee, S. (20 de 11 de 2008). *CLIPS Basic Reference Guide.* Obtenido de CLIPS Basic Reference Guide: http://www.csie.ntu.edu.tw/~sylee/courses/clips/bpg/top.html

Microsystems, S. (01 de 10 de 2008). *Sun Microsystems (API del paquete de sonido de JAVA).* Obtenido de Sun Microsystems: http://java.sun.com/javase/6/docs/technotes/guides/sound/

Microsystems, S. (01 de 10 de 2008). *Sun Microsystems (Diagrama de clases de Javax Sound).* Obtenido de Sun Microsystems (Diagrama de clases de Javax Sound): http://www.falkhausen.de/en/diagram/spec/javax.sound.html

Orós, C. (12 de 05 de 2008). *La Web de Carlos System.* Obtenido de La Web de Carlos System: http://www.carlosys.com/web/

Valle, C. P. (02 de 11 de 2008). *Sonido en Java.* Obtenido de Sonido en Java: http://cprades.eresmas.com/Tecnica/sonidoenjava.html

9. ANEXOS

9.1. ANEXO A - GLOSARIO DE TÉRMINOS

- **Armadura**: Es la parte que "define" la estructura de un pentagrama. En ella se encuentra la velocidad, la clave, o el tipo de composición.
- **Armonía**: Se conoce como armonía a la relación entre los diferentes sonidos musicales para crear una determinada composición.
- **Escala musical**: Dícese de un conjunto de notas con una estructura "interválida" determinada, es decir, con una distancia determinada entre cada una de las notas que la forman.
- **Intervalo**: Un intervalo en armonía musical es la distancia que separa a dos notas musicales. Los intervalos dotan de un carácter concreto a una composición musical y la teoría de intervalos es la base de la armonía moderna.
- **Melodía**: Conjunto de notas secuenciales simples (no acordes) que forman una composición musical.
- **Pentagrama**: Un pentagrama es un sistema de representación que se compone de cinco líneas horizontales (y 1 o más adicionales por arriba y por abajo) que sirve para representar una composición musical.
- **Tempo**: En la terminología musical, el tempo (Palabra en italiano que significa movimiento temporal) es la velocidad o el ritmo que se le da a una obra, es decir la frecuencia en el tiempo de los golpes de expresión sonora o rítmica de la composición. Es un elemento del sonido extremadamente crucial ya que afecta a la emotividad y dificultad de la pieza. *(FUENTE: Wikipedia)*

9.2. ANEXO B – CÓDIGOS DE MENSAJES DE VOZ MIDI

CÓDIGOS DE MENSAJES DE VOZ					
Byte de estado	Byte de datos 1	Byte de datos 2	Mensaje	Leyenda	Descripción
1000nnnn	0kkkkkkk	0vvvvvvv	**Nota OFF**	**n** = canal **k** = nota # 0-127 (60=middle C) **v** = velocity (0-127)	Se utiliza para **detener una nota**. El byte de estado indica el canal, el primer byte de datos la nota que se va a apagar y el segundo la velocidad, que corresponde a la velocidad con que se suelta la tecla
1001nnnn	0kkkkkkk	0vvvvvvv	**Nota ON**	**n** = canal **k** = nota # 0-127 **v** = velocity (0-127)	Se utiliza para **activar una nota**. El byte de estado indica el canal, el primer byte de datos la nota que se va a encender (el código 60 es el Do central) y el segundo la velocidad, que corresponde con la velocidad con que se pulse la tecla y que, según el instrumento con que se trabaje, puede suponer un incremento de la intensidad o no.
1010nnnn	0kkkkkkk	0ppppppp	**After Touch (Presión de las teclas)**	**n** = canal **k** = nota # 0-127(60=middle C) **p** = pressure (0-127)	Se tiene en cuestión la presión de la tecla a la hora de reproducir el sonido.
1011nnnn	0ccccccc	0vvvvvvv	**Control de cambio**	**n** = channel **c** = controller **v** = controller value(0-127)	Los veremos posteriormente, son los controladores.
1100nnnn	0ppppppp	[none]	**Cambio de programa**	**n** = channel **p** = preset number (0-127)	Cambia el programa del instrumento, es decir, cambia el sonido (guitarra, piano, viola, etc.)
1101nnnn	0ppppppp	[none]	**Canal de presión**	**n** = channel **p** = pressure (0-127)	
1110nnnn	0ccccccc	0fffffff	**Pitch Bend (Bending)**	**n** = channel **c** = coarse **f** = fine (c+f = 14-bit resolution)	Desplazan la altura tonal de la nota hacia arriba o hacia abajo. Sirve para desafinar el sonido, simulando así el estiramiento de las cuerdas de una guitarra, o similar

9.3. ANEXO C – NÚMERO DE CONTROLADOR MIDI ASIGNADOS

DECIMAL	HEX	NOMBRE DEL CONTROLADOR
1	**01h**	**CONTROL DE MODULACIÓN**
7	**07h**	**VOLUMEN PRINCIPAL**
10	**0Ah**	**CONTROL DE BALANCE (PAN)**
64	**40h**	**PEDAL DE TRANSPORTACIÓN (Sustain) [Data Byte of 0-63=0ff, 64-127=On]**
65	41h	TRANSPORTACIÓN
66	**42h**	**SOSTENUTO**
67	43h	Soft Pedal
68	44h	Legato Footswitch
69	45h	Hold 2
70	46h	Sound Controller 1 (default: Sound Variation)
71	47h	Sound Controller 2 (default: Timbre/Harmonic Content)
72	48h	Sound Controller 3 (default: Release Time)
73	49h	Sound Controller 4 (default: Attack Time)
74	4Ah	Sound Controller 5 (default: Brightness)
80-83	50-53h	General Purpose Controllers (Nos. 5-8)
84	**54h**	**CONTROL DE TRANSPORTACIÓN**

9.4. ANEXO D – OUTPUT DE PRUEBA 4.3

```
         CLIPS (Quicksilver Beta 09/24/07)
CLIPS>        (batch        "C:/Documents        and        Settings/Javi/Mis
documentos/Formación/UC3M/musicgenoma/bin/ejecucion_nivel_3.bat")
TRUE
CLIPS> (close archivo)
FALSE
CLIPS> (clear)
CLIPS> (load "../src/funciones_init.clp")
!!!!!!!
TRUE
CLIPS> (load "../src/representacion.clp")
%%%%%%%%
TRUE
CLIPS> (load "../src/especificaciones.clp")
$$$
TRUE
CLIPS> (load "../src/file_output.clp")
!!*!*
TRUE
CLIPS> (load "../src/inicializacion.clp")
$$$***$$*****
TRUE
CLIPS> (load "../src/funciones.clp")
!!!!!!!!
TRUE
CLIPS> (load "../src/reglas_control.clp")

[CSTRCPSR1] Expected the beginning of a construct.
***
FALSE
CLIPS> (load "../src/reglas_patron.clp")
***
TRUE
CLIPS>
(reset)
CLIPS> (run)
Nota A46CLIPS> (load "../src/reglas_insercion.clp")
*****
TRUE
CLIPS> (run)
Se incrementa contador de notas por compas a 2
Nota B.4.16 | Posición de compás: 48 | Número de compás: 1 | Tipo de nota:
inestable
Se incrementa contador de notas por compas a 3
Nota G.3.16 | Posición de compás: 64 | Número de compás: 1 | Tipo de nota:
semiestable

Cambiando contador de notas por compas a 3

Se incrementa contador de notas por compas a 1
Nota G.4.32 | Posición de compás: 32 | Número de compás: 2 | Tipo de nota:
estable
```

```
Se incrementa contador de notas por compas a 2
Nota D.4.32 | Posición de compás: 64 | Número de compás: 2 | Tipo de nota:
semiestable

Cambiando contador de notas por compas a 2

Se incrementa contador de notas por compas a 1
Nota G#.4.32 | Posición de compás: 32 | Número de compás: 3 | Tipo de nota:
semiestable
Se incrementa contador de notas por compas a 2
Nota B.4.16 | Posición de compás: 48 | Número de compás: 3 | Tipo de nota:
estable
Se incrementa contador de notas por compas a 3
Nota D#.4.16 | Posición de compás: 64 | Número de compás: 3 | Tipo de nota:
estable

Cambiando contador de notas por compas a 3

Se incrementa contador de notas por compas a 1
Nota B.4.16 | Posición de compás: 16 | Número de compás: 4 | Tipo de nota:
semiestable
Se incrementa contador de notas por compas a 2
Nota C.4.16 | Posición de compás: 32 | Número de compás: 4 | Tipo de nota:
estable
Se incrementa contador de notas por compas a 3
Nota G.4.32 | Posición de compás: 64 | Número de compás: 4 | Tipo de nota:
semiestable

Cambiando contador de notas por compas a 3



**********  INICIO  DE  IMPRESION  DE  PATRON  TRANSFORMADO  O+
**********

Patron: A Contador de notas de patron 1 Numero de notas de patron 3Elemento
recogido: C532
Patron: A Contador de notas de patron 2 Numero de notas de patron 3Elemento
recogido: B516
Patron: A Contador de notas de patron 3 Numero de notas de patron 3Elemento
recogido: G416

********** FIN DE IMPRESION DE PATRON **********


********** HE LLLEGADO **********


Cambiando contador de notas por compas a 0



********** INICIO DE IMPRESION DE PATRON TRANSFORMADO O+ **********
```

```
Patron: B Contador de notas de patron 1 Numero de notas de patron 2Elemento
recogido: G532
Patron: B Contador de notas de patron 2 Numero de notas de patron 2Elemento
recogido: D532

********** FIN DE IMPRESION DE PATRON **********


********** HE LLLEGADO **********


Cambiando contador de notas por compas a 0


**********      INICIO      DE      IMPRESION      DE      PATRON
**********

Patron: C Contador de notas de patron 1 Numero de notas de patron 3Elemento
recogido: G#432
Patron: C Contador de notas de patron 2 Numero de notas de patron 3Elemento
recogido: B416
Patron: C Contador de notas de patron 3 Numero de notas de patron 3Elemento
recogido: D#416

********** FIN DE IMPRESION DE PATRON **********


********** HE LLLEGADO **********


**********      INICIO      DE      IMPRESION      DE      PATRON
**********

Patron: D Contador de notas de patron 1 Numero de notas de patron 3Elemento
recogido: B416
Patron: D Contador de notas de patron 2 Numero de notas de patron 3Elemento
recogido: C416
Patron: D Contador de notas de patron 3 Numero de notas de patron 3Elemento
recogido: G432

********** FIN DE IMPRESION DE PATRON **********


********** HE LLLEGADO **********


**********      INICIO      DE      IMPRESION      DE      PATRON
**********

Patron: A Contador de notas de patron 1 Numero de notas de patron 3Elemento
recogido: C432
```

```
Patron: A Contador de notas de patron 2 Numero de notas de patron 3Elemento
recogido: B416
Patron: A Contador de notas de patron 3 Numero de notas de patron 3Elemento
recogido: G316

********** FIN DE IMPRESION DE PATRON **********


********** HE LLLEGADO **********


**********      INICIO      DE      IMPRESION      DE      PATRON
**********

Patron: B Contador de notas de patron 1 Numero de notas de patron 2Elemento
recogido: G432
Patron: B Contador de notas de patron 2 Numero de notas de patron 2Elemento
recogido: D432

********** FIN DE IMPRESION DE PATRON **********


********** HE LLLEGADO **********


**********  INICIO  DE  IMPRESION  DE  PATRON  TRANSFORMADO  O+
**********

Patron: C Contador de notas de patron 1 Numero de notas de patron 3Elemento
recogido: G#532
Patron: C Contador de notas de patron 2 Numero de notas de patron 3Elemento
recogido: B516
Patron: C Contador de notas de patron 3 Numero de notas de patron 3Elemento
recogido: D#516

********** FIN DE IMPRESION DE PATRON **********


********** HE LLLEGADO **********


Cambiando contador de notas por compas a 0


**********  INICIO  DE  IMPRESION  DE  PATRON  TRANSFORMADO  O+
**********

Patron: D Contador de notas de patron 1 Numero de notas de patron 3Elemento
recogido: B516
Patron: D Contador de notas de patron 2 Numero de notas de patron 3Elemento
recogido: C516
```

```
Patron: D Contador de notas de patron 3 Numero de notas de patron 3Elemento
recogido: G532

********************** FIN DE IMPRESION DE PATRON ************************


********************** HE LLLEGADO ************************


Cambiando contador de notas por compas a 0

FINAL 1
```

9.5. ANEXO E - MANUAL DE USUARIO

REQUISITOS

Los requisitos para la instalación son los siguientes:

- CLIPS 5.x o 6.x
- Eclipse 3.x
- Compilador de Java

GENERACIÓN DE LA MELODÍA CLIPS

Con la herramienta CLIPS generaremos un archivo con un formato de intercambio con el compilador java que luego generará el archivo MIDI.

Para realizar esto se deben seguir los siguientes pasos:

1. **Selección de las especificaciones**. Para seleccionar las especificaciones debemos editar el fichero contenido en la carpeta src/ llamado especificaciones.clp con la herramienta CLIPS o cualquier editor de texto sin formato, siendo un formato correcto de especificaciones el siguiente:

```
(deffacts especificaciones_compilador
        (tonica C)
        (modo menor)
        (timbre 8)
        (pulso 60)
        (armadura
                (duracion 4)  ;; 4 golpes por compás
                (numero 2)    ;; de duración negra (2) 1-Blanca 2-Negra 3-Corchea
        )
)
(deffacts especificaciones_generador_patrones
        (margenOctavas 2 4)
        (margenDuracionNotas 16 32)
        (margenSilencios 15 25) ;; especifica el margen de probabilidad de silencio / nºnotas
        (margenProgresionCromatica 20 30)
)

(deffacts estructura_melodia
        (numeroDeCompases 20)
        (estructuraMelodia (idPatron A B C B C B3+ C3+ D E D E B C D E )
)
```

teniendo en cuenta que:

a. La tónica puede ser entre las letras A y G, que se corresponde con las notas musicales en notación americana.
b. El modo se tiene que corresponder con la cadena *mayor* o *menor*.
c. En `(margenOctavas 2 4)` los dos números deben comprender el 1 y el 8 y el primer número debe ser menor o igual que el segundo.

d. En `(margenDuracionNotas 16 32)` los dos números deben comprender el 1 (semi-fusa) y el 64 (redonda) siendo el primer número menor o igual que el segundo.

e. En `(margenSilencios 15 25)` se especifica el intervalo en el que se pueden encontrar silencios, por ejemplo, en el caso que se expone, se podría dar un silencio entre cada 15 y 25 notas.

f. En el caso de los patrones

```
(numeroDeCompases 10)
(estructuraMelodia (idPatron A B C A3+ B3+ D E D E A)
```

las posibles combinaciones que se pueden utilizar para la estructura de la melodía son:

- Cualquier letra de la A a la Z

- Cualquier letra X ya utilizada con los modificadores XO+ y XO- (octava arriba o abajo), X3+ y X3- (Tercera arriba o abajo)

- El número de compases debe corresponderse con el número de símbolos de la estructura de la melodía.

2. **Ejecución en CLIPS y generación de la melodía**. Para generar la melodía, una vez modificadas las especificaciones, debemos abrir el CLIPS y realizar las siguientes acciones:
   a) File > Load Batch…
   b) Seleccionamos el archivo ejecucion_nivel_3.bat, que se encuentra en la carpeta bin del proyecto.
   c) Esto generará un archivo llamado melodía.pddl, que será el archivo de entrada para el programa JAVA

TRANSFORMACIÓN A MIDI

Para transformar la melodía a formato MIDI, debemos tener como entrada el archivo melodía.pddl, que se genera ya en el directorio adecuado.

Para generar la melodía se debe ejecutar el archivo Generador.java, que generará el archivo MIDI melodía.mid.